Algorithms and efficiency of Network calculus

Anne Bouillard

Algorithms and efficiency of Network calculus

Algorithmes et efficacité du Network calculus

par

Anne Bouillard

Mémoire pour l'obtention de

l'Habilitation à Diriger des Recherches

de l'École normale supérieure

(Spécialité Informatique)

Soutenue le 8 avril 2014 devant le jury composé de:

M. Laurent Hardouin, *Professeur*
M. Jean-Yves Le Boudec, *Professeur*
M. Sylvain Lombardy, *Professeur*
M. Ludovic Noirie, *Ingénieur de Recherche*
M. Marc Pouzet, *Professeur*
M. Jens Schmitt, *Professeur, Dr.-Ing.*
M. Lothar Thiele, *Professeur*

au vu des rapports de:

M. Jean-Yves Le Boudec, *Professeur*
M. Sylvain Lombardy, *Professeur*
M. Jens Schmitt, *Professeur, Dr.-Ing.*

Contents

1 Introduction **1**

2 The Network calculus model **5**
2.1 The (min,plus) dioid . . . 5
2.2 Data processes . . . 6
2.3 One type of arrival curves . . . 6
2.4 System guarantees and different types of service curves . . . 7
2.5 Performance guarantees . . . 8

3 Curves in network calculus **11**
3.1 Single-Input-Single-Output (SISO) systems . . . 11
3.2 Multiple-Input-Multiple-Output (MIMO) systems . . . 13
3.3 Composition of service curves . . . 15
3.4 Packet curves . . . 17

4 Performance bounds in networks **19**
4.1 Network of systems . . . 19
4.2 Loss of the tightness . . . 20
4.3 Performance bounds with (min,plus) techniques . . . 21
4.4 Tight performance bounds with linear programming . . . 22

5 Algorithms for (min,plus) functions **29**
5.1 A stable class for the Network calculus operators . . . 29
5.2 Fast convolution by a convex function . . . 31
5.3 Application to fast weak-KAM integrators . . . 34

6 Adapting the constraints to a flow **37**
6.1 Constraints of a flow . . . 37
6.2 Computation of the long-term behavior of a flow . . . 38
6.3 Application to the supervision of a flow . . . 41

7 Conclusion and perspectives **43**

Chapter 1

Introduction

With the emergence of a multitude of network architectures, performance evaluation, originating from Erlang's work, has become an important research topic. The simplest model of a communication system is the *queue*, and by extension systems of queues, that have a high modeling power. A queue is composed of a *waiting queue* and of a *server* that processes data in the waiting queue. If $X(t)$ is the number of data in the queue at time t, C the number of data that can be processed by unit of time, and $A(t)$ the number of data entering the system between tome t and $t+1$, the first (and most important) formula that can be written is the *Lindley formula* [84],

$$X(t+1) = \max(X(t) - C, 0) + A(t). \tag{1.1}$$

The queuing theory, based on this formula, derives properties for $X(t)$ based of the knowledge of $C(t)$ and $A(t)$ given by some distributions and independence relations. Among the important results derived from this formula are the Pollaczek-Khinchine formula and the Little formula [85]. As far as stochastic models are concerned, queuing theory is still an active topic, and with the emergence of large and complex networks new models have also being developed. Among them, one can mention the mean-field theory [8, 78, 66, 56], stochastic geometry [5, 6], random graphs [63, 13, 54]...

It is also possible to make use of the '+' and the 'max' in Equation (1.1). Then systems of queues can be analyzed using the (max,plus) (or tropical) algebra [7]. Compared to classical algebra, the addition is replaced by the maximum, and the multiplication by the addition. *Timed Petri nets* is a model that can handle some queuing networks - it naturally models concurrency, synchronicity and choices - and makes use of the (max,plus) algebra to derive performances of the system. When only concurrency and synchronization are involved, its boils down to the study of a (max,plus) matrix (with possibly random coefficients) and to the study of the eigen-value (or Lyapunov coefficient) of this matrix. when choices are involved, 1-bounded systems can be modeled by *heaps of pieces* [67, 68] or *(max,+) automata* [77, 76].

Another theory based on the + and the max in Equation (1.1) is the *Network calculus*, which is the subject of this document. Unlike *timed Petri nets* and classical queuing theory, it is based on the study of *envelopes* and bounding processes rather than studying their exact value. Introduced by the seminal work of Cruz [51, 52], in which the traffic is characterized by (σ, ρ)-envelopes to compute maximum delays. The notion of envelope has then been formalized and generalized to functions with values in the $(\min, +)$-dioid [53]. The elements of network, namely, the wires, switches, processors... have also been generalized from conservative link (the amount of data that can be served during each unit of time is constant) to more general envelopes in the same functional space. In this model, data flows and systems are abstracted by functions in the dioid of the (min,plus) functions and performances can be derived by combining those functions through (min,+) operators such as the (min,+)-convolution, the (min,plus) deconvolution or the sub-additive closure. The main references on the topic are the textbooks [45] and [79].

The target application was originally communication networks, such as Internet. Network calculus was successfully applied to study networks with differentiated services (DiffServ), as it enables to compute a guaranteed rate for the best effort flows and integrated services (IntServ), that guarantees a bandwidth for each flow [89, 90, 61]. Another success is the application of network calculus to define efficients load-balanced switches, the Birkhoff-Von Neumann switch for example, that has a periodic scheme to connect

any input to any output during a time proportional to the bandwidth requested for this connection [46, 47, 48]. A third example of application in this field is the video-on-demand (VoD) [87, 55, 64].

However, in other fields of communication networks, Network calculus leads to over-pessimistic results. Indeed, the transmission times are often not critical: worst-case performances are not the right parameter to study - it happens very rarely - whereas the mean or the variability of the transmission delay might be the parameters to study, as the users of the network might be more sensitive to a *slower than usual* network. As a consequence, stochastic models are more relevant. To this aim, the stochastic counterpart of Network calculus has been developed, the *stochastic network calculus* [44, 72, 60]. The aim of this theory is to compute the violation probability of some flow to have a certain maximum delay. As a consequence, it mixes the network calculus theory with the deviation theory.

Another class of applications where network calculus is relevant is real-time and critical systems. Those systems have hard deadlines and requires a deterministic analysis. An emblematic example where network calculus has been successfully used in the AFDX (Avionic Full Duplex), [62, 35, 40] and recent developments of network calculus have been obtained in this context. AFDX is an embedded network based on the Ethernet technology and where switches are connected using full-duplexed links, that is, between two switches, there are two different wires, one for each direction, avoiding collisions. A realistic network is composed of a dozen of switches and thousands of flows, called *virtual links*. As a consequence, the techniques developed to analyze such networks must be algorithmically efficient and compute accurate upper bounds on the transmission delays.

In the field of embedded networks, network calculus competes with other techniques. Among them, one can cite *model checking* [49]. Model checking is based on the exhaustive modeling of the states of the system with objects such as timed automata (or recently adapted to the context of network calculus, event-count automata [43]) and computes the exact bounds by analyzing them. As a consequence, it will give very accurate bounds, but at a prohibitive algorithmic cost. For example, in [92] a 3-node network can be analyzed in 30 minutes... A second and classical technique is scheduling. In the context of the AFDX network, the *trajectorial approach* has been developed [86]. Given a sporadic flow (almost periodic with jitters) and a packet of this flow, the aim is to find a bound on the worst-case delay suffered by this packet given the interacting flows. The equation giving this worst-case delay can then be solved using a fix-point equation. Here again, the bounds computed with this technique seem more accurate (though not exact) but the computing cost can be high. Unfortunately, flows have been found in this theory, invalidating it until further investigations [75, 74]. Instead, with network calculus techniques, the bounds may be less accurate, but the techniques used are computationally very light. Recent development have been made in order to tighten the bounds by taking into account a refined model of AFDX, for example by detailing the behavior of flows with fixed-size packets [102, 103].

As a consequence, there are several reasons in favor of using network calculus for those kinds of networks. Its nice modeling based on the (min,plus) algebra makes it modular (it is possible to analyze parts of the whole system and then see those parts as network elements), mathematically sound, and the computations are algorithmically efficient. Roughly speaking, network calculus is the combination of three concepts: it is designed for performance evaluation, it is theoretically based on the (min,plus) algebra and it uses a model of envelopes (or curves). Its originality, compared with performance evaluation using (max,plus) techniques such as Petri nets is the use of the curves, enabling non-determinism in the trajectories. Nevertheless, it suffers from two major limitations: the pessimism of the bounds as the size of the network grows and the inherent difficulties of the model. Indeed, as the theory was developed, the elegant (min,plus) framework was not enough to obtain good performance bounds and new concepts have been included. Among them, different types of service curves, packetization, residual service curves... On the one hand, system are described in a more and more detailed manner (beyond the classical arrival curve and service curve, the size of the longest packet, the type of the service curve, the service policy...) that require a modeling that is not only in the (min,plus) framework; on the other hand, the analysis is still performed using (min,plus) methods. Then, one can ask whether there is a possible modeling of the different concepts that have been introduced with (min,plus) functions or envelopes or if the (min,plus) algebra is the right framework to perform the final analysis. In the first part of the manuscript (Chapters 3 and 4), we will try to answer those questions.

In the second part of the manuscript, we will ask quite the reverse question, about using the network calculus concept (curves and (min,plus) functions) for other goals than performance evaluation. Of course, when dealing with tropical algebra the answer is yes: it is a branch of mathematics and is studied

in areas ranging from algebraic geometry (see [42] for a clear introduction) to analysis ([41]). Concerning the concept of curves, it is used for modeling clocks in synchronous systems [93], but the goal is to bound buffers, therefore, the aim is very close to that of performance evaluation. We will present one application for each of these points, where the solutions, have a network calculus flavor: the first one is the prolongation of [34], as defining an efficient algorithm for (min,plus) convolution, while the second one borrows directly the concept of arrival curves.

Organization of the document

More precisely, the main contributions are organized as follows.

Chapter 3 **Formalization of the network calculus concepts:** While attempting to compute efficient bounds, we faced the difficulty of handling different, but almost similar concepts of service curves, that model the guarantee of the servers in a network. After some beginner's mistakes, it became necessary to clarify those concepts and, if possible, unify them. The main contribution is a detailed comparison between the types of service curves presented in the literature. Here, we will present this comparison only between the two main types of service curves: the *simple service curve* and the *strict service curve*. The results can be set this way: **strict and simple service curves are inherently different and one cannot get rid of one to perform a global analysis.**

Chapter 4 **Computing tight bounds in networks:** the final objective of network calculus is to compute good performance upper bounds in networks. It is well-known that using (min,plus)-convolution lead to pessimistic performance bounds, and this pessimism become huge as the network grows. The main contribution is the computation of *exact* worst-case performance bounds for some networks, by getting rid of the (min,plus) algebraic framework and using linear programming instead. Under general assumptions, **blind multiplexing and FIFO networks can be modeled by linear constraints and computing the exact worst-case performance bounds boils down to solving a linear program. Moreover, this problem is NP-hard.**

Chapter 5 **Efficient algorithms for the basic operations:** in order to have efficient algorithms, the (min,plus) operators should also be efficiently implemented. Such algorithm have already been presented in [34]. Here, we present a **linear time algorithm to compute the (min,plus) convolution of a convex function by a concave function** and an application to numerical approximation of the long-term behavior of Hamilton-Jacobi equation.

Chapter 6 **Using network calculus constraints for supervising a flow:** In the context of the PhD of Aurore Junier, we present an algorithm to supervised flow. The key element of this supervision is the use of arrival curves, and we define **an on-line algorithm that follows the changes in the periodic behavior of a flow** and detect the changes of arrival rates, thus allowing some tolerance.

Chapter 2 is an introduction to Network calculus and Chapter 7 is the conclusion and presents some perspectives.

Other research works not described in this manuscript

Besides the content of this manuscript, I also got interest in other related models, that mostly concern the qualitative properties of some networks, whereas Network Calculus mainly concerns their quantitative properties. However, similar tools can be used, in the sense that they are based on the (min,plus) and (max,plus) algebra, through (max,plus) automata or Petri nets.

- **Timed Petri nets and web services:** Petri nets are a powerful tool to model concurrent timed systems and one example of such systems is web services. Those systems can be interpreted as services that are composed of elementary service through the operations of synchronization, concurrency and choice, which makes Petri nets a good modeling candidate. I was interested in the problem of the monotony of web-services and the identification of the *critical* elementary service. Monotonic web services (web-service whose response time is improved if the response time of any of

its elementary service is improved) are those that can be modeled with *event graphs* (they involve no choice, only concurrency and synchronization). Under Markovian assumptions, the critical service (the improvement of the response time of a critical elementary service improves the response time of the global service by the same quantity) can be identified by analyzing a Markov chain. This work, presented in [32] and [21], has been done in collaboration with Sydney Rosario, Albert Benveniste and Stefan Haar.

- **Residuation in (max,plus) automata:** Residuation of (max,plus) automata may have some application in control theory: when a system's timed behavior is not known exactly and some actions are controllable, a controller can force the timed behavior into a target behavior. The question is how to compute this controller. One way to do this is to model the frame the behavior of the original system by (min,plus) and (max,plus) automata, as well as the target behavior. The residuation of the system automata by the target automata will result in the controller automata. Then the question of computing this residuation raises. In fact, this is a difficult problem in general that is related to that of the determinization of (max,plus) automata [76]. But in our case, we avoid this difficulty by mixing (max,plus) and (min,plus) automata and computing the residuation of a (max,plus) by a (min,plus) automata. This work has been done in collaboration with Philippe Darondeau, Philippe Badouel and Jan Komenda and appears in [9].

- **Study of link-state routing protocols:** In the context of Aurore Junier's PhD and of the partnership of INRIA and Alcatel-Lucent, we studied the detection of anomalies in routing protocols such as OSPF ([70]). Besides the work presented in Chapter 6, we presented a refined modeling of part of the protocol by way of Petri nets. This modeling enables to modify some parameters of the protocols in order to simulate it and estimate the best parameters for the protocol. Further, we studied the correlation of infrequent alarms. Those works are presented in [22] and [29].

Chapter 2

The Network calculus model

In this chapter, we present the principal features of Network calculus, that is the use of the (min,plus) algebra and the use of envelopes to model the arrival processes. We first introduce in Section 2.1 the (min,plus)–framework on which are based the basic concepts of Network calculus and then start with the simplistic example of a single queue in Section 2.2 that is used to define the core notions in NC, the *curves*. Arrival curves is detailed in Sections 2.3 and service curves in Section 2.4. Finally, we present how to compute performance bounds from these curves in Section 2.5. Most of the content of this chapter can be found in the two reference books [45, 79].

2.1 The (min,plus) dioid

Let $\mathbb{R}_{\min} = \mathbb{R} \cup \{+\infty\}$. Equipped with the operators min and $+$, $\mathbb{R}_{\min}$ is an idempotent semi-ring, hence a *dioid*. It can be lifted to the dioid of the (min,plus) functions: we denote by $\mathcal{F}$ the set of functions from $\mathbb{R}_+$ to $\mathbb{R}_{\min}$ and $\mathcal{F}_\uparrow$ the set of functions in $\mathcal{F}$ that are non-decreasing, left-continuous and take value 0 at 0, that is $\mathcal{F} = \{f : \mathbb{R}_+ \to \mathbb{R} \cup \{+\infty\}\}$ and $\mathcal{F}_\uparrow = \{f : \mathbb{R}_+ \to \mathbb{R} \cup \{+\infty\} \mid f(0) = 0$ and f is left-continuous and non-decreasing$\}$. Then $(\mathcal{F}, \wedge, *)$ and $(\mathcal{F}_\uparrow, \wedge, *)$ are dioids with unit element $\delta_0 : 0 \mapsto 0;\ t \mapsto \infty$ otherwise and zero element, $\varepsilon : t \mapsto \infty$ and where $\forall f, g \in \mathcal{F}$,

- **point-wise minimum:** $\forall t \in \mathbb{R}_+,\ f \wedge g(t) = \min(f(t), g(t))$ and
- **(min,plus)-convolution:** $\forall t \in \mathbb{R}_+,\ f * g(t) = \inf_{0 \le s \le t} f(s) + g(t-s)$.

In addition to those two operators, two others can be defined: the *sub-additive closure*, that plays the role of a pseudo-inverse, and the *deconvolution* that can also be viewed as the equivalent as the residuation (or pseudo-division) operator in the context of dioids.

Sub-additive closure Let $f \in \mathcal{F}$. Define $f^0 = \delta_0$ the unit element and $f^{n+1} = f * f^n$. Then, $\forall f \in \mathcal{F}$,

- **sub-additive closure:** $f^* = \bigwedge_{n \in \mathbb{N}} f^n$.

For f^* to be well-defined, f must be non-negative in a neighborhood of 0 (otherwise, f^* could take $-\infty$ values). So, if $f \in \mathcal{F}_\uparrow$, then f^* is well-defined and, in addition, belongs to $\mathcal{F}_\uparrow$. An alternative is to use the *complete* semi-ring, $\overline{\mathbb{R}_{\min}} = \mathbb{R}_{\min} \cup \{-\infty\}$. The function f^* is also defined as the largest sub-additive function that is not larger than f.

This operator is useful to compute the performances of systems with backward control.

Deconvolution: a pseudo-division Let $f, g, h \in \mathcal{F}$. We have the equivalence

$$f * g \ge h \Leftrightarrow \forall 0 \le s \le t,\ f(s) + g(t-s) \ge h(t) \Leftrightarrow$$
$$\forall 0 \le s \le t,\ f(s) \ge h(t) - g(t-s) \Leftrightarrow \forall t \ge 0,\ f(t) \ge \sup_{u \ge 0} h(t+u) - g(u).$$

This equivalence defines the *residuation* of the convolution operator.

- **(min,plus) deconvolution:** $\forall f, g \in \mathcal{F}$, $\forall t \in \mathbb{R}_+$, $f \oslash g(t) = \sup_{u \geq 0} f(t+u) - g(u)$.

This operator will be used when computing the worst-case backlog ($\alpha \oslash \beta(0)$) and the propagation of the arrival constraints ($\alpha \oslash \beta$) for a system offering a service curve β to a CAF that is α-upper constrained. More algebraic properties about the deconvolution can be found in [79].

2.2 Data processes

Flows and servers (or systems) in the network are modeled by left-continuous, non-decreasing functions $f : t \mapsto f(t)$, where t represents *time* and $f(t)$ a *quantity of data*. In the whole document, unless stated otherwise, we will assume that the time is *continuous* and $t \in \mathbb{R}_+$.

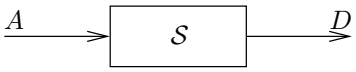

Figure 2.1: A system $\mathcal{S}$ with cumulative arrival function A and cumulative departure function D.

Consider the example represented on Figure 2.1: one system is crossed by one data flow and during any period of time t this system can serve exactly $R.t$ bits of data. Let $A(t)$ be the quantity of data that *arrives* at the system until time t and $D(t)$ be the amount of data that *departs* from the system until time t. Initially, the system is empty and we choose $A(0) = 0$ and $D(0) = 0$. The function A is called the *cumulative arrival function* (CAF) of the system and D the *cumulative departure function* (CDF) of the system. Let us now derive the relation between A and D.

Suppose that during the time interval $]u, t]$, the system is never empty (we also say that it is always backlogged), meaning that during this period of time, exactly $R(t-u)$ bits of data exit the system:

$$D(t) - D(u) = R(t-u). \tag{2.1}$$

The assumption that the system is always backlogged is mandatory. Otherwise we would have $D(t) - D(u) \leq R(t-u)$, which is not a guarantee for the service offered.

As $A(0) = D(0)$, $s_0 = \sup\{s \leq t \mid A(s) = D(s)\}$ the last instant before t at which $A(s_0) = D(s_0)$ exists. Using the latter formula, we obtain

$$D(t) = A(s_0) + R(t - s_0).$$

If $s < s_0$, then $D(t) \leq D(s) + R(t-s) \leq A(s) + R(t-s)$ as $D(s) \leq A(s)$.
If $s > s_0$, then $A(s) + R(t-s) \geq A(s) + D(t) - D(s) \geq D(t)$ and we finally obtain

$$D(t) = \inf_{0 \leq s \leq t} A(s) + R(t-s). \tag{2.2}$$

This formula corresponds to the (min,plus)-convolution of A and $\beta : t \mapsto R.t$: in short, we note $D = A * \beta$.

Using this formula, performance bounds can be derived when A is known. For example, the backlog at time t is $A(t) - D(t) = A(t) - \inf_{0 \leq s \leq t} A(s) + R(t-s) = \sup_{0 \leq s \leq t}(A(t) - A(s)) - R(t-s)$. The aim of network calculus is to compute worst-case bounds when A and the amount of service offered are not known exactly, using some knowledge about them. This knowledge is given by *arrival and service curves* that we now define.

2.3 One type of arrival curves

Definition 1 (Arrival curve)**.** *A CAF $A \in \mathcal{F}_\uparrow$ is* upper-constrained *by $\alpha \in \mathcal{F}$ if $\forall s \leq t$,*

$$A(t) - A(s) \leq \alpha(t-s).$$

A CAF $A \in \mathcal{F}_\uparrow$ is lower-constrained *by $\underline{\alpha} \in \mathcal{F}$ if $\forall s \leq t$,*

$$A(t) - A(s) \geq \underline{\alpha}(t-s).$$

The functions α and $\underline{\alpha}$ are respectively called maximal *and* minimal arrival curves *of A.*

We will use the following notations: $\mathcal{A}(\alpha) = \{A \in \mathcal{F}_\uparrow \mid \forall 0 \leq s \leq t,\ A(t) - A(s) \leq \alpha(t-s)\}$ is the set of CAFs that are upper-constrained by α and $\underline{\mathcal{A}}(\underline{\alpha}) = \{A \in \mathcal{F}_\uparrow \mid \forall 0 \leq s \leq t,\ A(t) - A(s) \geq \underline{\alpha}(t-s)\}$ is the set of CAFs that are lower-constrained by $\underline{\alpha}$.

In network calculus, the minimal arrival curves are often omitted, but it can be useful, in particular when the CAF is known to be quite regular. When the minimum or maximum nature of an arrival curve is not mentioned, it is by default a maximum arrival curve.

A common class of arrival curves are the affine functions: $\alpha : t \mapsto \sigma + \rho t$, where σ represents the maximum amount of data that can arrive simultaneously and ρ the maximal long-term arrival rate.

It is well-known that the maximal arrival curves can be replaced by their sub-additive closure. Indeed, $\forall s \leq u \leq t$, $A(t) - A(s) = A(t) - A(u) + A(u) - A(s) \leq \alpha(t-u) + \alpha(u-s)$, and α can be chosen such that $\alpha(u+v) \leq \alpha(u) + \alpha(v)$ for all $u, v \geq 0$. We denote by α^* the sub-additive closure of α. Similarly, $\underline{\alpha}$ can be chosen super-additive. These closures are optimal in the sense that $\forall \alpha' < \alpha^*$, $\mathcal{A}(\alpha') \subsetneq \mathcal{A}(\alpha)$.

When dealing with both α and $\underline{\alpha}$, the question of whether the choice of those functions is optimal is more involving and has been addressed by Altisen and Moy in [3, 4]. They present an algorithm that computes the smallest $\alpha' \leq \alpha$ and the greatest $\underline{\alpha}' \geq \underline{\alpha}$ such that $\mathcal{A}(\alpha') \cap \underline{\mathcal{A}}(\underline{\alpha}') = \mathcal{A}(\alpha) \cap \underline{\mathcal{A}}(\underline{\alpha})$ for large classes of functions.

2.4 System guarantees and different types of service curves

Definition 2 (System)**.** *$\mathcal{S} \subseteq \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow$ is called a system, or server, if $\forall (A, D) \in \mathcal{S}$, $A \geq D$. In other words, it is a binary relation between the CAFs and the CDFs of this system.*

We will always assume that there is no loss and no creation of data in the system. Then the constraint that $A \geq D$ is natural. Also note that this relation may not be defined for all the CAFs and is not deterministic (one CAF may produce several CDFs). Given $\mathcal{S} \subseteq \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow$, $(A, D) \in \mathcal{S}$ is called a *trajectory* of $\mathcal{S}$.

A guarantee on the service offered allows to define constraints on the CDF, given a CAF. Following our example, we obtain two equivalent definitions for a constant rate server:

$$D = A * \beta \quad \text{and} \quad D(t) - D(u) = \beta(t-u) \text{ if } \forall s \in]u, t[,\ A(s) > D(s).$$

Unfortunately, those two definitions are not equivalent in the general case and both are required in order to deal with networks of systems.

Definition 3 (Simple service curve)**.** *A system $\mathcal{S} \subseteq \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow$ offers the minimum simple service curve (or guarantees the service) β if*

$$\mathcal{S} \subseteq \mathcal{S}_{simple}(\beta) = \{(A, D) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow \mid A \geq D \geq A * \beta\}.$$

A system $\mathcal{S} \subseteq \mathcal{F} \times \mathcal{F}$ is said to offer the maximum service curve $\overline{\beta}$ if

$$\mathcal{S} \subseteq \overline{\mathcal{S}}(\overline{\beta}) = \{(A, D) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow \mid 0 \leq D \leq \min(A * \overline{\beta}, A)\}.$$

In network calculus, the maximal service curves is often omitted, but it can be useful to improve the performance bounds computed. Moreover, a usual maximum service curve is given by the capacity of the *links* of the network. When the minimum or maximum nature of a service curve is not mentioned, it is by default a minimum service curve.

Given a trajectory $(A, D) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow$, a *backlogged period* is an interval $I \subseteq \mathbb{R}_+$ of time during which the backlog is non-null, *i.e.* $\forall u \in I$, $A(u) > D(u)$. For $t \in \mathbb{R}_+$, the *start of the backlogged period* of t is $start(t) = \sup\{u \leq t \mid A(u) = D(u)\}$. Since the cumulative functions A and D are assumed left-continuous, it holds $A(start(t)) = D(start(t))$. If $A(t) = D(t)$, then $start(t) = t$. For any $t \in \mathbb{R}_+$, $]start(t), t[$ is a backlogged period (and so is $]start(t), t]$ if $A(t) > D(t)$).

Definition 4 (Strict service curve)**.** *A system $\mathcal{S} \subseteq \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow$ offers the strict service curve β if*

$$\mathcal{S} \subseteq \mathcal{S}_{strict}(\beta) = \{(A, D) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow \mid A \geq D, \text{ and } \forall \text{ backlogged period }]s, t], D(t) \geq D(s) + \beta(t-s)\}.$$

Three common families of service curves are

- the *pure delay* curve $\delta_T : t \mapsto 0$ if $t \in [0, T]$; $t \mapsto +\infty$ otherwise. When simple service curves are considered, this means that the sojourn time of each bit of data is at most T: $A * \delta_T = A((\cdot - T)_+)$, where $(x)_+ = \max(x, 0)$. When strict service curve are considered, this means that a backlogged period never exceeds T.
- the *guaranteed rate* curve $\lambda_R : t \mapsto Rt$. For every arrival process, the minimal admissible trajectory is such that either the service rate is exactly ρ or the server is empty. This is the service curve of our example. The minimal admissible trajectory is the same for strict and simple service curves. However, the difference is that during a backlogged period, the service rate R is not guaranteed with a simple service curve, whereas it is with a strict service curve.
- the *rate-latency* service curve $\beta : t \mapsto R(t - T)_+$ that is the combination of the two latter curves. Given a CAF, the minimum CDF is such that every bit of data first waits a time T, and among the bits of data that waited T units of time, the service rate is R for simple service curves. For strict service curves, for each service curve, there is a latency of T where no data is served, then data is served at rate R, whenever it arrives.

2.5 Performance guarantees

Given an input/output system, bounds for the worst-case backlog and worst-case delay can be easily deduced from the arrival and service curves.

Consider a system $\mathcal{S}$ and (A, D) a trajectory of $\mathcal{S}$. The *backlog* of the flow at time t is $b(t) = A(t) - D(t)$, and the delay endured by data entering at time t (assuming FIFO discipline for the flow) is

$$\begin{aligned} d(t) &= \inf\{s \geq 0 \mid A(t) \leq D(t+s)\} \\ &= \sup\{s \geq 0 \mid A(t) > D(t+s)\}. \end{aligned}$$

For the trajectory, the *worst-case backlog* is $B_{\max} = \sup_{t \geq 0} \big(A(t) - D(t)\big)$ and the *worst-case delay* is $D_{\max} = \sup_{t \geq 0} d(t) = \sup\{t - s \mid 0 \leq s \leq t \text{ and } A(s) > D(t)\}$.

For the system S, the *worst-case backlog* (resp. *delay*) is the supremum over all its trajectories. Upper bounds for these worst-case performances can be computed using the following theorem.

Theorem 1 ([45, 79])**.** *Let $\mathcal{S}$ be a system and $\alpha, \beta \in \mathcal{F}$ such that $\mathcal{S} \subseteq \mathcal{S}_{simple}(\beta)$. If $(A, B) \in \mathcal{S}$ and $A \in \mathcal{A}(\alpha)$, then*

1. *$B_{\max} \leq \sup\{\alpha(t) - \beta(t) \mid t \geq 0\} = \alpha \oslash \beta(0)$ (maximal vertical distance between α and β).*
2. *$D_{\max} \leq \inf\{d \geq 0 \mid \forall t \geq 0,\ \alpha(t) \leq \beta(t+d)\}$ (maximal horizontal distance between α and β).*
3. *$\alpha' = \alpha \oslash \beta$ is an arrival curve for D (or $D \in \mathcal{A}(\alpha \oslash \beta)$).*

The worst-case backlog is bounded by the maximal vertical distance between α and β while the worst-case delay is given by the maximal horizontal distance between those two functions. Figure 2.2 illustrates this fact. Those bounds are *tight* if α is sub-additive ([79]) and $\beta(0) = 0$, that is, there exists $(A, B) \in \mathcal{S}(\beta)$ such that $B_{\max} = \alpha \oslash \beta(0)$ and $D_{\max}$ is the horizontal distance between α and β: for example, it is obtained with $A = \alpha$ and $B = \min(\beta, \alpha) (\geq \alpha * \beta)$.

The constraint propagation can be improved in the case where a minimal arrival curve $\underline{\alpha}$ and a maximal service curve $\overline{\beta}$ are known for the flow and the system:

$$\alpha' = (\alpha * \overline{\beta}) \oslash \beta \qquad \text{and} \qquad \underline{\alpha}' = \underline{\alpha} * \beta.$$

2.5.1 Backward control

This operator can be use for backward control of a system: consider a system offering a service curve β, and that the input is controlled so that the amount of data in the system never exceeds W. The complementary data is stored in another queue, waiting to enter the system. This system is depicted in Figure 2.3 (left). With the notations of the figure, we obtain a service guarantee for the global system (input/output relation between A and B) $\beta_1^{eff} = \beta * (\beta + W)^*$:

$$[A_1 = A \wedge B + W = A \wedge B * I_w \wedge B \geq A_1 * \beta] \Rightarrow B \geq (A \wedge B * I_W) * \beta \Rightarrow B \geq A * \beta * (\beta + W)^*,$$

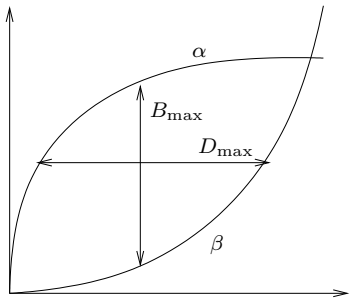

Figure 2.2: Guaranteed bounds on backlog and delay.

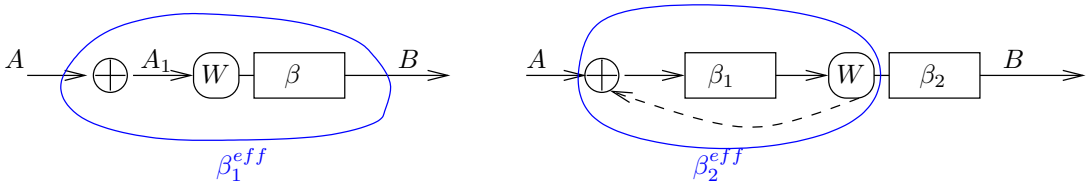

Figure 2.3: Backward control of a queue. Left: a simple system; right: application to write-block systems with a finite buffer.

with $I_W : 0 \mapsto W; t \mapsto +\infty$ otherwise. Note that $\forall f \in \mathcal{F}$, $f * I_W = f + W$.

As show in [31], this model can also be used to compute the exact service offered by systems in tandem, when the first system can be blocked when the the buffer of the second system is full, as shown on Figure 2.3 (right). In this case, the service offered by the first system, taking into account the blocking due to the second system is

$$\beta_2^{eff} = \beta_1 * ((\beta_2 + W) * \beta_1)^*.$$

This results can be generalized to tandem of arbitrary size [31] and to fork-join networks [104].

Chapter 3

Curves in network calculus

Arrival and service curves are necessary to compute performance guarantees in Network calculus, but service curves have received much more attention as it is a much trickier concept and as a system can be described more precisely than by a service curve only. First, as suggested in the previous chapter, several types of service curves co-exist, and the necessity of this must be explained and clarified. Second, we only defined systems with a single input and a single output (SISO), whereas most of the time, systems have multiple inputs and multiple outputs (MIMO). Then the notion of system must be extended to such systems for which the service policy must also be taken into account. This enables us to preciser bounds, leading to the notion of *residual service curve*. Third, the data that flows in the system is made of packets in practice, which slightly modifies the departure process if the whole packet is assumed to depart at once from the system. Usually, this issue is tackled by taking into account the size of the longest packet in the service curve. However, a more precise description of the sizes of the packets may be given by *packet curves*, that counts the number of entire packets in a given amount of data. For some service policies, it can lead to more accurate service curves.

Following these three points, this chapter is organized as follows: In Section 3.1, we compare simple and strict service curves. In Section 3.2, we introduce the notion of MIMO server and describe different service policies and the corresponding *residual service curves*, before presenting the composition of service curves, which will allow to compute performance bounds for composite networks in Section 3.3. The chapter ends by defining *packet curves* in Section 3.4 and shows some examples of use cases. The work about the comparison of service curves has been partially done with Laurent Jouhet and Éric Thierry, while the work about packet curves has been done with Nadir Farhi and Bruno Gaujal.

3.1 Single-Input-Single-Output (SISO) systems

Dealing with strict and simple service curves is mandatory when analyzing composite systems and we address here the question of whether it is possible to find an equivalence between those two notions in order to make it possible to deal with only one type of curve. Unfortunately, the answer will be no. To show this, let us first focus on systems with one input and one output flow.

In this section, for the sake of simplification, we will only deal with service curves in $\mathcal{F}_\uparrow$. In [24, 14], these results have been presented in the more general context of $\mathcal{F}$ and to more types of service curves defined below.

3.1.1 Different types of service curves

In the literature, different notions of service curves can be found. For the sake of simplicity, we will only deal with the two main types of service curves, but let us comment on some other possible definitions.

- ***Weakly strict service curve:*** $\mathcal{S}_{wstrict}(\beta) = \{(A, D) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow \mid A \geq D, \text{ and } \forall t \geq 0,\ D(t) \geq D(start(t)) + \beta(t - start(t))\}$. This notion is studied in [23, 24] and is intermediate between simple and strict service curves. Some results from the strict service curves can be adapted to this case.
- ***Variable capacity node:*** $\mathcal{S}_{vcn}(\beta) = \{(A, D) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow \mid \exists C \in \mathcal{F}_\uparrow, \forall t \geq 0,\ D(t) = \inf_{0 \leq s \leq t} \big[A(s) + C(t) - C(s)\big] \text{ and } \forall 0 \leq s \leq t,\ C(t) - C(s) \geq \beta(t - s)\}$. This notion has been widely used in the

special case of the guaranteed rate server $C(t) - C(s) = R(t-s)$. In most of the cases, the notion of variable capacity nodes and strict service curves coincide. An important family of service curves where the equivalence does not hold is the pure delays $\delta_T, T \in \mathbb{R}_+$.

- ***Adaptive service curve:*** $\mathcal{S}_{asc}(\beta, \tilde{\beta}) = \{(A, D) \in \mathcal{S} \times \mathcal{S} \mid A \geq D \text{ and } \forall t \in \mathbb{R}_+,\ \forall s \leq t,\ D(t) \geq (D(s) + \tilde{\beta}(t-s)) \wedge \inf_{s \leq u \leq t} A(u) + \beta(t-u)\}$. This notion has been defined in [1] in order to take into account the advantages of the simple and of the strict service curves. Dealing with two different curves however introduces additional complexity and no evidence that better performance bounds can be derived has been found.

- ***Adaptive service curve (unique curve):*** $\mathcal{S}_{uasc}(\beta) = \{(A, D) \in \mathcal{S} \times \mathcal{S} \mid A \geq D \text{ and } \forall t \in \mathbb{R}_+,\ \forall s \leq t,\ D(t) \geq (D(s) + \beta(t-s)) \wedge \inf_{s \leq u \leq t} A(u) + \beta(t-u)\}$. When the two curves are equal in the adaptive service curve, it can be shown (in [14]) that when β is convex, $A * \beta \in \mathcal{S}_{uasc}(\beta)$, so in this case, better performance bounds cannot be derived.

Real-time calculus The notion of service curve in the *real-time calculus* theory (RTC) is slightly different from the previous ones: besides the fact that maximum service curves and minimal arrival curves are systematically used, there are two main differences:

- the time variable ranges over $\mathbb{R}$ and not $\mathbb{R}_+$;
- CAF and CDF are defined as functions of two parameters: $A(s,t)$ is the amount of data that arrives during the interval $[s,t)$.

One can easily get rid of the second difference, as the cumulative processes obeys to the Chasles relation ($A(u,s) + A(s,t) = A(u,t)$) and then $A(s,t)$ can be replaced by $A(t) - A(s)$ in the network calculus theory. The first difference is fundamental as the performance bounds obtained with RTC are different (see [105] and [107]) for example). The main reason of this is that the worst-case performances are often obtained near the initialization point $t = 0$ in network calculus. In RTC, this point does not exist. Real-time calculus when time ranges over $\mathbb{R}_+$ is exactly the variable capacity nodes.

3.1.2 Hierarchy and equivalence

Let us now give a precise comparison between simple and strict service curves.

First, the notion of service curve is monotonic for non-decreasing curves: let $\beta, \beta' \in \mathcal{F}_\uparrow$. We have the following equivalences:

- $\mathcal{S}_{simple}(\beta) \subseteq \mathcal{S}_{simple}(\beta') \Leftrightarrow \beta \geq \beta'$;
- $\mathcal{S}_{strict}(\beta^{\bar{*}}) \subseteq \mathcal{S}_{strict}(\beta'^{\bar{*}}) \Leftrightarrow \beta^{\bar{*}} \geq \beta'^{\bar{*}}$,

where $\beta^{\bar{*}}$ is the super-additive closure of β (the smallest super-additive function that is greater than β).

Second, the following well-known theorem, establishes a hierarchy between the strict and simple service curves.

Theorem 2. *Let $\beta \in \mathcal{F}_\uparrow$. Then $\mathcal{S}_{strict}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$. Moreover, $\mathcal{S}_{strict}(\beta) = \mathcal{S}_{simple}(\beta) \Leftrightarrow \beta_\uparrow = \delta_0$ or 0.*

Indeed, for any $t \in \mathbb{R}_+$, set $s = start(t)$, the last start of backlogged period before t. For $(A, D) \in \mathcal{S}_{strict}(\beta)$, we can write $D(t) \geq D(s) + \beta(t-s) = A(s) + \beta(t-s) \geq A * \beta(t)$.

There are only two cases where the equality between strict and simple service curves holds: the null service curve and the infinite service curve. Nevertheless, if one considers the equality of the output process when the service is exact (that is, when the last inequalities are replaced by an equality in the definition of $\mathcal{S}_\mathcal{T}(\beta)$, $\mathcal{T} \in \{simple, strict\}$) for any arrival process, equivalence cases are more frequent: for example when $\beta = \lambda_r, r \in \mathbb{R}_+ \cup \{+\infty\}$.

The following theorem states that there cannot exist an equivalence between simple and strict service curves apart from the two cases mentioned above.

Theorem 3. *1.* **Non equivalence of service types :** *Let $\beta \in \mathcal{F}_\uparrow$. If $\beta \neq \delta_0$ and 0, then $\nexists \beta' \in \mathcal{F}_\uparrow$ such that $\mathcal{S}_{simple}(\beta') = \mathcal{S}_{strict}(\beta)$.*

2. **Families of curves :** *Let I and J be finite sets and $(\beta_i)_{i\in I}$ and $(\beta'_j)_{j\in J}$ be two families in $\mathcal{F}_\uparrow$. Then, $\bigwedge_{i\in I}\mathcal{S}_{simple}(\beta_i) = \bigwedge_{j\in J}\mathcal{S}_{simple}(\beta'_j)$ if and only if $\{\beta\in\mathcal{F} \mid \exists i\in I,\ \beta\le\beta_i\} = \{\beta\in\mathcal{F}\mid \exists j\in J,\ \beta\le\beta'_j\}$.*

3. **No translation with families :** *$\nexists (\beta_i)_{i\in I}\in\mathcal{F}_\uparrow^I,\ (\beta'_j)_{j\in J}\in\mathcal{F}_\uparrow^J,\ \bigwedge_{i\in I}\mathcal{S}_{simple}(\beta_i) = \bigwedge_{j\in J}\mathcal{S}_{strict}(\beta'_j)$, except for the equality cases defined in Theorem 2.*

For the first assertion, β can be assumed to be super-additive. If there existed $\beta'\in\mathcal{F}_\uparrow$ such that $\mathcal{S}_{simple}(\beta') = \mathcal{S}_{strict}(\beta)$, then we would have $(\delta_0,\beta)\in\mathcal{S}_{simple}(\beta')$ and so $\beta'\le\beta$. But, also $(\delta_0,\beta')\in\mathcal{S}_{strict}(\beta)$, so $\beta'\ge\beta$, hence $\beta'=\beta$.

The second assertion states that even families of simple service curves cannot be expressed in a simpler way: assume (without loss of generality due to the monotony property) that the curves in $(\beta_i)_{i\in I}$ are not 2-by-2 comparable and neither are the curves in $(\beta'_j)_{j\in J}$. In order to have the equivalence of the two families of service curves, there must be a one-to-one correspondence ϕ between I and J such that $\beta_i = \beta'_{\phi(i)}$. Strict service curves behave better concerning this issue: if $(\beta_i)_{i\in I}$ be a family of super-additive functions in $\mathcal{F}_\uparrow$ (which can always be assumed), $\bigwedge_{i\in I}\mathcal{S}_{strict}(\beta_i) = \mathcal{S}_{strict}(\sup_{i\in I}\beta_i)$..

Finally, the third assertion generalizes the first one to families of curves.

3.2 Multiple-Input-Multiple-Output (MIMO) systems

Let us now focus on systems with multiple input and output flows. The worst-case performances of a single flow can be computed if the service effectively offered to the flow, which we call the *residual* service, is known. This section presents how this residual service is computed under various service policies using Network calculus. As mentioned above, one must care about the type of service curve involved.

MIMO as the aggregation of flows

A relation $\mathcal{S}$ is a system with m input flows and m output flows if $\mathcal{S}\subseteq\mathcal{F}_\uparrow^m\times\mathcal{F}_\uparrow^m$ and satisfies $\forall((A_i)_{i=1}^m,(D_i)_{i=1}^m)\in\mathcal{S}$, $\forall i\in\{1,\dots,m\}$, $A_i\ge D_i$. The aggregated system if $\mathcal{S}$ is

$$Ag(\mathcal{S}) = \{(\sum_{i=1}^m A_i, \sum_{i=1}^m D_i) \mid ((A_i)_{i=1}^m,(D_i)_{i=1}^m)\in\mathcal{S}\}$$

and the projection of this server on $I\subseteq\{1,\dots,m\}$ is $P_I(\mathcal{S}) = \{((A_i)_{i\in I},(D_i)_{i\in I}) \mid ((A_i)_{i=1}^m,(D_i)_{i=1}^m)\in\mathcal{S}\}$. Those notations are illustrated on Figure 3.1.

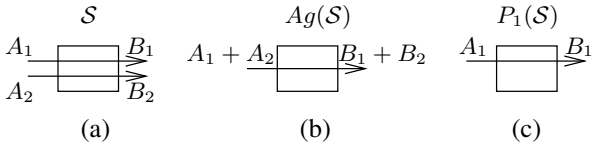

Figure 3.1: (a) system with two input and two output flows; (b) aggregated system; (c) projection of the system on the first input/output.

The system $\mathcal{S}\in\mathcal{F}^m\times\mathcal{F}^m$ offers a service curve β of type $\mathcal{T}$ if $Ag(\mathcal{S})\in\mathcal{S}_\mathcal{T}(\beta)$. In the following, we only consider the case $m=2$, from which the general case can be deduced. We will always assume that each flow has a FIFO service policy.

Theorem 4 (Blind multiplexing, [79])**.** *Let $\mathcal{S}\in\mathcal{F}_\uparrow^2\times\mathcal{F}_\uparrow^2$ and $\beta\in\mathcal{F}_\uparrow$. The following statements hold.*

(i) If $\exists T>$ and $\beta(T)=0$, $Ag(\mathcal{S})\subseteq\mathcal{S}_{simple}(\beta) \nRightarrow \exists\beta'\neq 0$ such that $P_1(\mathcal{S})\subseteq\mathcal{S}_{simple}(\beta')$;

(ii) $Ag(\mathcal{S})\subseteq\mathcal{S}_{strict}(\beta)$ and $P_2(\mathcal{S})\subseteq\mathcal{A}(\alpha_2) \nRightarrow P_1(\mathcal{S})\subseteq\mathcal{S}_{strict}((\beta-\alpha_2)_\uparrow)$.

$\mathcal{S}\subseteq\mathcal{A}(\alpha)$ is an abuse of notation. It stands for $\{A\mid(A,B)\in\mathcal{S}\}\subseteq\mathcal{A}(\alpha)$.

Concerning (ii), it is possible to obtain a residual strict service curve as stated in [14], but the residual curve computed, $(\beta-\alpha_2\oslash(\beta-\alpha_1)_\uparrow)_\uparrow$ if A_1 is α_1-upper constrained, would be too pessimistic to obtain good bounds on the delay, so we will not use it.

In Theorem 4, we made no assumption about the service policy and the worst-case scenario in considered. This is *arbitrary*, or *blind* multiplexing. More precise results exist when dealing with a precise service policy, and we now present some of them.

Different service policies

We consider a system crossed by 2 flows, flow 1 and flow 2, with respective CAFs A_1 and A_2 and CDFs D_1 and D_2.

First In First Out (FIFO) Data is served in the order of its arrival, independently of the flow it belongs to. This means that a bit of a flow arriving at time t will be transmitted only when all the traffic arrived before time t (and belonging to any flow traversing that server) has been transmitted. More formally, a system $\mathcal{S} \in \mathcal{F}_\uparrow^2 \times \mathcal{F}_\uparrow^2$ is FIFO if for all $(A_1, A_2, D_1, D_2) \in \mathcal{S}$, $\forall i, j \in \{1, 2\}$, $\forall u, t \in \mathbb{R}_+$,

$$\begin{aligned} D_i(t) > A_i(u) &\Rightarrow D_j(t) \geq A_j(u) \text{ and} \\ D_i(t) < A_i(u) &\Rightarrow D_j(t) \leq A_j(u). \end{aligned}$$

Network Calculus also allows one to derive *equivalent service curves* for the individual flows, which can then be employed to compute delay bounds through Theorem 1.

Proposition 1. *[79, Chapter 6.2] Let $\beta, \alpha_2 \in \mathcal{F}_\uparrow$ and $\mathcal{S} \subseteq \mathcal{F}^2 \times \mathcal{F}^2$ a FIFO system.*

$$\left.\begin{array}{l} Ag(\mathcal{S}) \in \mathcal{S}_{simple}(\beta) \\ P_2(\mathcal{S}) \in \mathcal{A}(\alpha_2) \end{array}\right\} \Rightarrow \begin{array}{l} \forall \theta \in \mathbb{R}_+,\ P_1(\mathcal{S}) \in \mathcal{S}_{simple}(\beta_\theta^1), \\ \text{with } \beta_\theta^1(t) = [\beta(t) - \alpha_2(t-\theta)]_+ 1_{t>\theta}. \end{array}$$

Note that those service curves are not comparable, and it is not possible to derive a unique equivalent service curve. If $\beta : t \mapsto R(t-T)_+$ is rate-latency and $\alpha_2 : t \mapsto \sigma_2 + \rho_2 t$ is affine, then β_θ^1 with $\theta = \frac{\sigma_2}{R} + T$ is also a rate-latency function and $\alpha_1 \oslash \beta_\theta^1$ is the smallest arrival curve for the CDF.

Static priorities Flows can be ordered according to the priority they have over the other input flows. Suppose that flow 1 is given a higher priority than flow 2. Then, if on the time interval $[s, t]$, flow 1 is always backlogged, flow 2 will not receive any service:

$$\forall u \in [s, t],\ A_1(u) - D_1(u) > 0 \Rightarrow D_2(t) = D_2(s).$$

This property can be used to show that the residual service is strict.

Proposition 2 ([23])**.** *Let $\beta, \alpha_1 \in \mathcal{F}$ and $\mathcal{S} \subseteq \mathcal{F}^2 \times \mathcal{F}^2$. Then, if flow 1 has a higher priority than flow 2,*

$$\left.\begin{array}{l} Ag(\mathcal{S}) \subseteq \mathcal{S}_{strict}(\beta) \\ P_1(\mathcal{S}) \subseteq \mathcal{A}(\alpha_1) \end{array}\right\} \Rightarrow \left\{\begin{array}{l} P_1(\mathcal{S}) \subseteq \mathcal{S}_{strict}(\beta) \text{ and} \\ P_2(\mathcal{S}) \subseteq \mathcal{S}_{strict}((\beta - \alpha_2)_\uparrow) \end{array}\right.$$

GPS (generalized processor sharing) Consider $\phi_1, \phi_2 \geq 0$ such that $\phi_1 + \phi_2 = 1$. The service policy is *general processor sharing* (GPS) if for any interval of time where flow $i \in \{1, 2\}$ is backlogged,

$$\phi_j(D_i(t) - D_i(s)) \geq \phi_i(D_j(t) - D_j(s)).$$

That is, each flow i is guaranteed a proportion ϕ_i of the total service offered by the system.

The following formula generalizes the work done by Pareck and Gallager in [89] where Proposition 3 is proved for guaranteed rates only.

Proposition 3. *Let $\mathcal{S} \in \mathcal{F}^2 \times \mathcal{F}^2$, $\beta, \alpha_1 \in \mathcal{F}_\uparrow$ and $\phi_1 \in (0, 1)$. Assume that β is convex and that the service policy is GPS, flow 1 being guaranteed a proportion ϕ_1 of the service and flow 2 a proportion $1 - \phi_1$.*

$$\left.\begin{array}{l} Ag(\mathcal{S}) \subseteq \mathcal{S}_{strict}(\beta) \\ P_1(\mathcal{S}) \subseteq \mathcal{A}(\alpha_1) \end{array}\right\} \Rightarrow P_2(\mathcal{S}) \subseteq \mathcal{S}_{strict}(\max[\phi_2\beta, (\beta - \alpha_1)_+]).$$

3.3 Composition of service curves

The composition of service curves is another very important aspect of Network calculus. Thank to the algebraic framework, composing simple service curves is very easy: it corresponds to the (min,plus)-convolution. Unfortunately, here again, the composition is not compatible with the other types of service curves, as stated in Theorem 5.

3.3.1 Two systems in tandem

We first consider two servers $\mathcal{S}_1$ and $\mathcal{S}_2$ in tandem, as represented on Figure3.2. The system representing the concatenation of those two servers is

$$\mathcal{S}_2 \circ \mathcal{S}_1 = \{(A, D_2) \mid \exists D_1 \in \mathcal{F} \text{ such that } (A, D_1) \in \mathcal{S}_1 \text{ and } (D_1, D_2) \in \mathcal{S}_2\}.$$

The composition of n similar systems is defined by $\mathcal{S}^0 = \{(A, A) \mid A \in \mathcal{F}_\uparrow\}$ and $\forall n \geq 0$, $\mathcal{S}^{n+1} = \mathcal{S} \circ \mathcal{S}^n$.

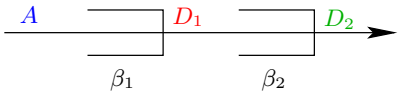

Figure 3.2: Servers in tandem.

Theorem 5. *The following statements hold.*

(i) $\mathcal{S}_{simple}(\beta_2) \circ \mathcal{S}_{simple}(\beta_1) \subseteq \mathcal{S}_{simple}(\beta_1 * \beta_2)$*;*

(ii) Let β_1 *and* β_2 *such that there exist* $T_1, T_2 > 0$ *with* $\beta_1(T_1) = 0$ *and* $\beta_2(T_2)$*=0. Then* $\nexists \beta > 0$ *such that* $\mathcal{S}_{strict}(\beta_2) \circ \mathcal{S}_{strict}(\beta_1) \subseteq \mathcal{S}_{strict}(\beta)$*.*

The proof of the classical statement *(i)* can be found in [79, 45]. Note that the inclusion may be strict as shown in [14], where the second statement is also proved.

3.3.2 A new notion of service curve?

We have just discussed two important properties of service curves: the computation of residual service curves and the composition of service curves. These properties respectively allow to deal with several flows and composite systems. On the one hand, we have to deal with strict service curves for computing residual service curves (with the notable exception of FIFO multiplexing), and on the other hand, the composition of service curves results in simple service curves. The existence of an intermediate type of service curve, between the simple and the strict, that would be stable with the composition and the residuation especially with the blind multiplexing) would simplify the theory. In this section, we explain why this not possible. The complete proof can be found in [14].

Assume that there exists such a notion of intermediate curve. More precisely, let us denote by $\hat{\mathcal{S}}(\beta)$ the relation between trajectories that must be satisfied for an intermediate service curve β. Then $\hat{\mathcal{S}}$ should satisfy at least

1. $\mathcal{S}_{strict}(\beta) \subseteq \hat{\mathcal{S}}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$;
2. $\hat{\mathcal{S}}(\beta_2) \circ \hat{\mathcal{S}}(\beta_1) \subseteq \hat{\mathcal{S}}(\beta_1 * \beta_2)$.

Let $\mathcal{S} \subseteq \mathcal{F} \times \mathcal{F}$. The *closure* of $\mathcal{S}$ is

$$\overline{\mathcal{S}} = \{(A, D') \in \mathcal{F} \times \mathcal{F} \mid \forall \varepsilon > 0,\ \exists (A, D) \in \mathcal{S} \text{ such that } \forall t \in \mathbb{R}_+,\ A(t) \geq D'(t) \geq D(t - \varepsilon)\}.$$

$\overline{\mathcal{S}}$ is the smallest system of $\mathcal{F} \times \mathcal{F}$ containing $\mathcal{S}$, closed and such that $(A, D) \in \overline{\mathcal{S}} \Rightarrow \forall D' \geq D$, $(A, D') \in \overline{\mathcal{S}}$. With this definition, the problem of the strict inclusion of the composition exposed in Theorem 5 is avoided: $\overline{\mathcal{S}_{simple}(\beta_2) \circ \mathcal{S}_{simple}(\beta_1)} = \overline{\mathcal{S}_{simple}(\beta_1 * \beta_2)}$, and we can state the theorem:

Theorem 6. *To each convex and piecewise affine function β in $\mathcal{F}$, associate $\hat{\mathcal{S}}(\beta)$ a system such that $\mathcal{S}_{strict}(\beta) \subseteq \hat{\mathcal{S}}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$. If $\forall \beta_1, \beta_2$ convex piecewise affine functions $\hat{\mathcal{S}}(\beta_2) \circ \hat{\mathcal{S}}(\beta_1) \subseteq \hat{\mathcal{S}}(\beta_1 * \beta_2)$, then $\forall \beta$, $\overline{\hat{\mathcal{S}}(\beta)} = \mathcal{S}_{simple}(\beta)$.*

To see the intuition of the proof, consider the case of pure delay service curves. Let $T > 0$ and consider the service curves δ_T and $\delta_{T/n}$. When the service is simple, we have $\overline{\mathcal{S}_{simple}(\delta_{T/n}) \circ \cdots \circ \mathcal{S}_{simple}(\delta_{T/n})} = \overline{\mathcal{S}_{simple}(\delta_T)}$. Let A be a cumulative arrival process in a system composed of n systems in tandem, offering a strict service curve $\delta_{T/n}$. When n grows to $+\infty$, and when the service is exact, the cumulative departure process tends to $A * \delta_T$, as illustrated on Figure 3.3.

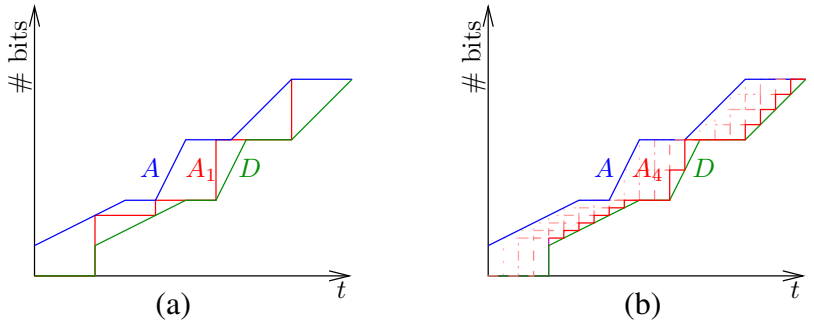

Figure 3.3: Example of CDF after 1 and 4 servers in tandem. (a) one server: the CAF A (blue) and CDF for strict A_1 (red) and simple D (green) service curves. (b) CAF A and CDF D for a simple service curve. From light to dark, the CDFs after each server with strict service curve $\delta_{T/4}$. A_1, A_2 and A_3 are dashed and A_4, the CDF of the whole system is plain.

Sufficiently strict service curves Recently Schmitt, Golan *et al.* in [97] introduced a new type of service curves in order to deal with non-FIFO flows and their performance evaluation. Those *sufficiently strict service curves* are defined as follows.

- $\mathcal{S}_{s3c}(\beta) = \{(A, D) \in \mathcal{F} \times \mathcal{F} \mid \forall t \in \mathbb{R}_+,\ A(t) \geq D(t) \geq A(t - Dw(t)) + \beta(Dw(t))\}$,

where $Dw(t)$ is the *maximum achievable dwell period* (MADP) at time t, that is, $D(t) = t - t_0(t)$ where t_0 is the arrival time of the oldest bit of data in the system at time t under *all possible processing orders*. Under our assumptions (FIFO per flow), and for a single server, $t_0(t) = start(t)$. In [97, Theorem 5], the authors show that their curve has the concatenation property, which seems in contradiction with Theorem 6, and appears as a good candidate for intermediate service curves. In fact, this is not, because of the parameter $Dw(t)$ that contains more information than just the service curve. Nevertheless, Theorem 6 suggests that the description of a system by a service curve is not precise enough to describe a system, and lots of information is lost by the concatenation and by computing individual service curves. Then, adding the information contained in the MADP may lead to the right notion: let $\mathcal{D} : \mathcal{S}_{s3c}(\beta) \to \mathcal{P}(\mathcal{F})$; $(A, D) \mapsto \mathcal{D}((A, D)) \subseteq \{Dw \in \mathcal{F} \mid Dw \text{ is a possible dwell period for } (A, D)\}$ (a *possible* dwell period must be such that $\forall t \in \mathbb{R}_+$, $Dw(t) \leq t - start(t)$). Then, one can define another type of service curves:

- $\mathcal{S}(\beta, \mathcal{D}) = \{(A, D) \in \mathcal{F} \times \mathcal{F} \mid \exists Dw \in \mathcal{D}(A, D),\ \forall t \geq 0,\ A(t) \geq D(t) \geq A(t - Dw(t))\}$.

Then the following theorem is straightforward: the first statement is a reformulation of [97, Theorem 5] and the second statement is obvious as $Dw(t) = t - start(t)$.

Theorem 7. *Let $\beta, \beta_1, \beta_2 \in \mathcal{F}$, $\mathcal{D} : \mathcal{S}_{s3c}(\beta) \to \mathcal{P}(\mathcal{F})$, $\mathcal{D}_1 : \mathcal{S}_{s3c}(\beta_1) \to \mathcal{P}(\mathcal{F})$ and $\mathcal{D}_2 : \mathcal{S}_{s3c}(\beta_2) \to \mathcal{P}(\mathcal{F})$ be possible dwell periods for respectively β,β_1 and β_2. Then,*

- *$\mathcal{S}_{s3c}(\beta_2, \mathcal{D}_2) \circ \mathcal{S}_{s3c}(\beta_1, \mathcal{D}_1) \subseteq \mathcal{S}(\beta_1 * \beta_2, \mathcal{D}')$ with $\forall t$, $\mathcal{D}'(A, C) = \{Dw \mid \exists Dw \in \mathcal{F}, \exists Dw_1 \in \mathcal{D}_1(A, D), Dw_2 \in \mathcal{D}_2(D, C),$ such that $Dw(t) = Dw_2(t) + Dw_1(t - Dw_2(t))\}$;*
- *If $\mathcal{S} \in \mathcal{F}^2 \times \mathcal{F}^2$, then $Ag(\mathcal{S}) \subseteq \mathcal{S}_{s3c}(\beta, \mathcal{D})$ and $P_2(\mathcal{S}) \subseteq \mathcal{A}(\alpha_2) \Rightarrow P_1(\mathcal{S}) \subseteq \mathcal{S}_{s3c}((\beta - \alpha_2)_\uparrow, \mathcal{D})$.*

This notion of service curve is an intermediate between simple and strict service curves, with the nice properties we targeted. But, if this curve is obtained after several steps of computations, the topology of the network being studied is hidden in $\mathcal{D}$. Moreover, up to our knowledge, there is no nice representation for the MADP.

3.4 Packet curves

We have now described the two classes of curves used in Network calculus and assumed that the flows are fluid: each packet can be split in pieces of infinitesimal size. However, in practice, this is not always the case and packets cannot be divided. An easy way to analyze packet flows is to study it as a fluid flow, but modifying the service curves to take into account the size of the packet. More precisely, if β is a strict service curve for a flow of packets whose size does not exceed $\ell_{\max}$, then $(\beta - \ell_{\max})_+$ is a strict service curve for the packetized flow.

The aim of this section is to present a possible analysis when more knowledge in available about the packet sizes, using the concept of *packet curves*. More details are presented in [17, 18].

Packet operator and packet curves In addition to a CAF A, one may consider P the *packet flow* associated to this CAF: $\forall t \geq 0$, $P(t)$ is the number of *entire* packets that arrive until time t. The transformation of an arrival flow into a packet flow is made using the *packet operator*, which is a function $\mathcal{P} : \mathbb{R} \to \mathbb{N}$ such that for an amount x of arrival data, $\mathcal{P}(x)$ is the number of *entire* packets in x: $P = \mathcal{P} \circ A(t)$.

The operator $\mathcal{P}$ may not be perfectly known, but some information about it may be available, more precise than only the minimum and maximum packet length respectively denoted by $\ell_{\min}$ and $\ell_{\max}$. For example, a flow with packets of size 1 and 2, where in three successive packets, there are at least one packet of size 1 and at least one packet on size 2. In order to take into account this information, we introduce the *packet curve* of a packet operator:

A function π (resp. Π) is a minimum (resp. maximum) packet curve for $\mathcal{P}$ if $\forall 0 \leq x \leq y$,

$$\mathcal{P}(y) - \mathcal{P}(x) \geq \pi(y - x) \quad (\text{resp. } \mathcal{P}(y) - \mathcal{P}(x) \leq \Pi(y - x)).$$

With the example described above, one can take $\pi : x \mapsto (3/5(x - 2/3))_+$ and $\Pi : x \mapsto 3/4x + 3/2$, where $(x)_+ = \max(0, x)$. Stair-case functions are more precise, but affine functions and rate-latency functions are easier to handle from an computational viewpoint. Note that if π is defined as $\pi : x \mapsto \mu(x - \nu)_+$, then $\nu \geq l_{\max}$ and $\mu \geq 1/\ell_{\max}$ and if Π is defined as $\Pi : x \mapsto V + Ux$, then $V \leq 1/\ell_{\min}$ and $V \geq 1$.

Properties of the packet curves We assume here that A (resp. A_i, $i \in \{1, 2\}$) is upper-constrained by the arrival curve α (resp. α_i) and has packet operator $\mathcal{P}$ (resp. $\mathcal{P}_i$) with packet curves π and Π (resp. π_i and Π_i) and that β is a (strict) service curve of the server. Then, the following properties hold.

(i) Π and π are maximal and minimal packet curves for D.

(ii) $\Pi \circ \alpha$ is an arrival curve for the packet flow P.

(iii) $\pi \circ \beta$ is a minimum (strict) service curve for P.

(iv) If β' be a minimum simple (resp. strict) service curve for a packet flow $P = \mathcal{P} \circ A$, then, $(\Pi)^{-1} \circ \lfloor \beta \rfloor$ (resp. $(\Pi)^{-1} \circ \lceil \beta \rceil$) is a minimum simple (resp. strict) service curve for $(\mathcal{P})^{-1} \circ \mathcal{P} \circ A$.

(v) $\pi_1 * \pi_2$ is a minimum packet curve for the (blind) aggregation of A_1 and A_2.

We now describe two examples where packet curves are useful.

Superposition of periodic flows. When the aggregation of several flow is FIFO, better packet curves than $\pi_1 * \pi_2$ can be found. A special case is the superposition of periodic flow. Consider N flows, where flow n, $1 \leq n \leq N$, is composed of packets of size S_n arriving with period T_n. Set $\rho_n = S_n/T_n$. Then π_N and Π_N are respective minimum and maximum packet curves for the superposition of those flows:

$$\pi(x) = \left(\sum_{n=1}^{N} \frac{x}{T_n \sum_i \rho_i} - \sum_{n=1}^{N} \frac{\sum_i T_i \rho_i}{T_n \sum_i \rho_i} \right)_+ \qquad \Pi(x) = \sum_{n=1}^{N} \frac{x}{T_n \sum_i \rho_i} + \sum_{n=1}^{N} \frac{\sum_i T_i \rho_i}{T_n \sum_i \rho_i}.$$

Note that the rates of the two functions are equal, thus optimal.

Non-preemptive service curves. Figure 3.4 gives the general scheme of computation, using the basic properties of packet curves. This scheme is more efficient that the computations that may be done

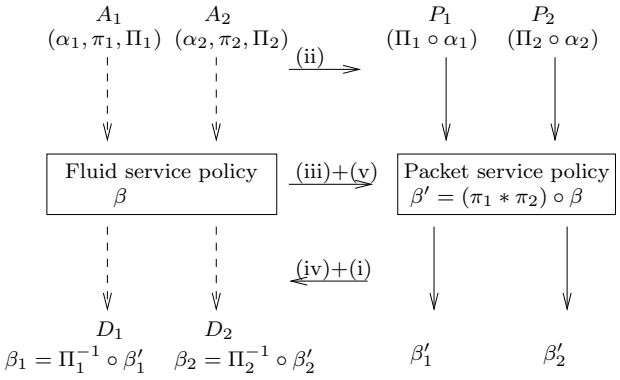

Figure 3.4: Non-preemptive service calculus scheme.

using classical methods and taking into account the minimum and maximum packet sizes only when the service policy is based on counting packets. An example of such policy is the Round-Robin. For this policy, three service curves can be computed (we consider two flows, and the residual service curve of the first):

- Classical method: $\beta_c(t) = \left(\frac{\ell_i^{\min}}{n\ell^{\max}} \beta - \ell^{\max} \right)^+$,
- Scheme method: $\beta_s(t) = (\Pi_i)^{-1} \left(\frac{1}{n} (\pi_i * \pi_2 \circ \beta) - 1 \right)^+$,
- Ad-hoc method: $\beta_{ah}(t) = (Id + \pi_2^{-1} \circ \Pi_1 + 1)^{-1} \circ (\beta - \ell_{\max})_+$.

Numerically, with $\ell_{\min} = 1$, $\ell_{\max} = 2$, $\pi_1 = \pi_2 = \pi$, $\Pi_1 = \Pi_2 = \Pi$ defined above, and $\beta(t) = 10t$, we have $\beta_c(t) = 2.5(t - 0.8)_+$, $\beta_s(t) = 4(t - 0.97)_+$ and $\beta_{ah}(t) = 4.44(t - 0.68)_+$. The residual service rate using the scheme is much better, but the ad-hoc method is the best.

In this chapter we presented the basic building blocks for computing a service curve for a given flow: for each flow and each server a residual service curve can be computed and then for these service curves can be composed to obtain a global service curve. The adaptations have also been presented in case the flow is made of packets. Then, performance bounds can be derived. The next chapter will present the advantages and limitations of such an approach and will be more specifically focused on the performance guarantees.

Chapter 4

Performance bounds in networks

In the previous chapter, we formalized the notion of server and defined operations that can be used to compute performance bounds in more complex networks by using the operation of composition of systems (convolution of service curves) and by computing residual service curves, depending on the type of service curve involved. We are now ready to compute performance bounds for more complex networks using these results. Although algorithmically efficient and elegant, these methods are over pessimistic: the tightness of the bounds, that is valid for one server, is lost as soon as at least two flows and two servers are involved. Then, an important issue is how to compute exact performance bounds and what is the complexity of this problem. We will answer those questions by using a different technique, linear programs.

In Section 4.1, we introduce the model and notations and in Section 4.2, we present a simple example illustrating the difficulties of getting exact bounds with purely (min,plus) techniques. Then, in Section 4.3, we give a general algorithm for computing performance bounds with those techniques before presenting in Section 4.4 a new method based on the mathematical programming, and more specifically on the linear programming. The main drawback of this method is its complexity that is exponential in the size of the network. In fact, the problem of computing a tight bounds under this framework is NP-hard. The blind multiplexing problem has been done in collaboration with Éric Thierry and Laurent Jouhet, while the FIFO problem is the result of a collaboration with Giovanni Stea.

4.1 Network of systems

Let us first define the notations for this chapter. They are quite heavy, however this is necessary in order to define the relations between the flows and the systems rigorously. We still use the following convention: systems are represented by letters h, k and appear as exponents. Flows are represented by the letters i and j, and appear as indices.

Description of a network Consider a network $\mathcal{N}$ is composed of n systems and m flows.

- **description of a flow:** $\forall i \in \{1, \dots, m\}$, flow i is described by an arrival curve α_i and a path $\mu_i = \langle \mu_i(1) \cdots \mu_i(\ell_i) \rangle$ where $\mu_i(\ell) \in \{1, \dots, n\}$. We write $h \in \mu_i$ or $i \in \mathrm{Fl}(h)$ if there exits $\ell \in \{1 \dots, \ell_i\}$ such that $h = \mu_i(\ell)$. If $h = \mu_i(\ell)$, then $\mathrm{pred}_i(h) = \mu_i(\ell - 1)$, with the convention that if $j = 1$, then $\mathrm{pred}_i(h) = 0$. We also note $\mathrm{end}(i)$ the last system visited by i, that is, $\mu_i(\ell_i)$.
- **description of a server:** $\forall h \in \{1, \dots, n\}$, system h is described by a service curve $\beta^{(h)} \in \mathcal{F}_\uparrow$ (that is strict or simple depending on the context).

Trajectories For $i \in \{1, \dots, m\}$, $h \in \mu_i$, we denote by $F_i^{(h)}$ the CDF of flow i from system h and by $F_i^{(0)}$ its CAF. The family of cumulative processes $(F_i^{(h)})_{i \in \{1,\dots,m\}, h \in \{0\} \cup \mu_i}$ is called a trajectory of the system. If moreover the following conditions are satisfied, it is called an *admissible* trajectory:

1. for each flow $i \in \{1, \dots, m\}$, $F_i^{(0)} \in \mathcal{A}(\alpha_i)$;

2. for each server $h \in \{1, \ldots, n\}$, $(\sum_{i \in \mathrm{Fl}(h)} F_i^{(\mathrm{pred}_i(h))}, \sum_{i \in \mathrm{Fl}(h)} F_i^{(h)}) \in \mathcal{S}_{\mathcal{T}}(\beta^{(h)})$, where $\mathcal{T} \in \{simple, strict\}$ depending on the context.

3. Each server h obeys to its service policy.

The underlying graph of the network is the graph with set of vertices $\{1, \ldots, n\}$ and set of edges $\{(k,h) \mid \exists i \in \{1, \ldots, m\}, \mu_i = \langle \ldots, k, h \ldots \rangle\}$. We will only consider feed-forward networks, whose underlying graphs is acyclic.

A system is said to be stable if there exists a constant C such that for each server, the backlog is always upper bounded by C. Let $R_h = \lim_{t \to \infty} \beta_h(t)/t$ and $\rho_p = \lim_{t \to \infty} \alpha_p(t)/t$. We assume that the system is stable, that is, $\forall h \in [1, N]$, $R_h \geq \sum_{p \ni h} \rho_p$ (see [79] for example).

4.2 Loss of the tightness

Given a network, we say that the performance bounds computed are tight if there exists an admissible trajectory of the network that have those performances. To illustrate the complexity of getting good bounds, consider the following small example with two servers and two flows.

- for $h \in \{1, 2\}$, $\beta^{(h)} : t \mapsto R^{(h)}(t - T^{(h)})_+$, flow 1 has a higher priority than flow 2;
- for $i \in \{1, 2\}$, $\alpha_i : t \mapsto \sigma_i + \rho_i$, $\mu_i = \langle 1, 2 \rangle$.

To compute an upper bound on the delay of flow 2, at least two methods can be used:

(a) first compute the residual service curves for flow 2 and compute the convolution of the two curves obtained: $\tilde{\beta}_a = (\beta^{(1)} - \alpha_1)_+ * (\beta^{(2)} - (\alpha_1 \oslash \beta^{(1)}))_+$.

(b) first compute the convolution of the two service curves and then compute the residual service for flow 2. Note that even though the service is not strict anymore and Theorem 4 can not be applied, a direct computation as performed in [23, 99, 100] leads to the residual service curve $\tilde{\beta}_b = (\beta^{(1)} * \beta^{(2)} - \alpha_1)_+$.

The second formula uses the principle of *pay multiplexing only once*, introduced in [100]: each bit of data of flow 2 can be overtaken only once by a bit of data of flow 1. This phenomenon is taken into account in the second computation only. Intuitively, this formula should lead to better delay bounds. In fact, that is not always the case. We find

$$\tilde{\beta}_a(t) = (\min(R_1, R_2) - \rho_1)(t - \frac{\sigma_1 + R_1 T_1}{R_1 - \rho_1} - \frac{\sigma_1 + \rho_1 T_1 + R_2 T_2}{R_2 - \rho_1})_+$$

and

$$\tilde{\beta}_b(t) = (\min(R_1, R_2) - \rho_1)(t - \frac{\sigma_1 + \min(R_1, R_2)(T_1 + T_2)}{\min(R_1, R_2) - \rho_1})_+.$$

If $\beta_1 = \beta_2$, then $\tilde{\beta}_2 \leq \tilde{\beta}_1$. But if $\sigma_1 = 0$, $T_1 = 0$ and $R_2 > R_1$, then $\tilde{\beta}_1 \leq \tilde{\beta}_2$.

The first method, used to compute $\tilde{\beta}_a$, can be generalized for general feed-forward networks. This is the object of Section 4.3. The second method, leading to $\tilde{\beta}_b$, can be generalized in two ways. First, for tandem networks, using a *multi-dimensional* (min,plus)-convolution, as defined in [19] and [20]. For example, a residual for flow 2 would be in that case $\forall t \geq 0$

$$\tilde{\beta}_c(t) = \min_{u+s=t} \beta^{(1)}(s) + \beta^{(2)}(u) - \alpha_1(s + u),$$

which leads in this specific case to the same expression as $\tilde{\beta}_b$. Unfortunately, there is no known algorithm to compute such a function in polynomial. But the horizontal and vertical distances with a concave function with the multi-dimensional (min,plus)-convolution of concave arrival curves and convex service curves can be performed in polynomial time (details of this approach can be found in [20] and [30]).

A second way to generalize this approach is to use linear programs. Indeed, it follows more or less the same computing scheme: first work with the trajectories, and bound using the arrival and service curves only at the last step of the computation. This will be discussed in Section 4.4.

4.3 Performance bounds with (min,plus) techniques

As we consider that the network is feed-forward: the servers can be numbered such that each path μ_i is increasing ($\mu_i(j) > \mu_i(j-1)$), which we now assume from now on.

Algorithm 1 gives a generic way to compute performance bounds. First, for each flow at each system it computes $\alpha_i^{(h)}$, an arrival curve for $F_i^{(h)}$, $\beta_i^{(h)}$, a residual service curve for flow i at system h. Then, it computes the global service curve for each flow by a convolution and worst-case performance upper bounds can be computed using that curve and Theorem 1.

Algorithm 1: Generic SFA (separated flow analysis)

Data: $(\alpha_i) \in \mathcal{F}_\uparrow^m$, $\beta_j \in \mathcal{F}_\uparrow^n$, $\mu \in \{1,\dots,n\}^m$, where μ_i is increasing.

Result: $\tilde{\beta}_i$ a simple service curve for each flow i, for the global network.

```
1 begin
2   for h = 1 to n do
3     foreach i such that h ∈ μ_i do
4       β_i^(h) ← (β^(j) − Σ_{i∈Fl(h)} α_i^(pred_i(h)))_+ ;
5       α_i^(h) ← α_i^(pred_i(h)) ⊘ β_i^(h) ;
6   foreach i = 1 to m do
7     β̃_i = ✻_{h∈μ_i} β_i^(h)
8 end
```

Algorithm 1 is valid for blind multiplexing (and thus for any other service policy) if the systems offer strict service curves. The performance bounds computed with this algorithm can be improved if more information is known about the systems and the curves.

1. If maximum service curves $\overline{\beta}^{(h)}$ and and minimum arrival curves $\underline{\alpha}_i$ are known for each system h and each flow i, then lines 4-5 can be replaced by the four following lines (with obvious notations):

$$\begin{array}{lcl} \beta_i^{(h)} & \leftarrow & (\beta^{(j)} - \sum_{i\in \mathrm{Fl}(h)} \alpha_i^{(\mathrm{pred}_i(h))})_+; \\ \alpha_i^{(h)} & \leftarrow & \min(\alpha_i^{(\mathrm{pred}_i(h))} * \overline{\beta}^{(\mathrm{pred}_i(h))} \oslash \beta_i^{(h)}, \overline{\beta}^{(\mathrm{pred}_i(h))}); \\ \overline{\beta}^{(h)} & \leftarrow & \overline{\beta}^{(\mathrm{pred}_i(h))}; \\ \underline{\alpha}_i^{(h)} & \leftarrow & \underline{\alpha}_i^{(\mathrm{pred}_i(h))} * \beta_i^{(h)}; \end{array}$$

2. If the service policy is FIFO (server can offer a simple service curve), SP or GPS, the it suffices to replace lines 4-5 by the formulas corresponding to the specific policy, that is using Propositions 1, 2 or 3. Naturally, it is possible to combine the different service policies with maximum service curves and minimum arrival curve.

The algorithm presented here does not claim for optimality, but for algorithmic efficiency. Indeed, it requires $\mathcal{O}(nm)$ basic operations (convolution, deconvolution...). When only one type of service policy is chosen for the whole network, it is possible to take advantage of this service policy.

For example, the FIFO policy has been extensively studied, specially by Lenzini, Mangozzi and Stea. From Proposition 1, an infinite number of service curves can be computed, and we have to choose $\theta^{(h)}$ for each server h that fits the best. A classical solution, when dealing with rate-latency service curves and affine arrival curves is to take $\theta^{(h)} = \frac{\sigma}{R} + T$, so that the residual service curves are still rate-latency. This choice may not be the best anymore when several servers in tandem are considered. The choice of these $\theta^{(h)}$ has received quite a lot of attention, leading to the implementation of a software, Deborah (*Delay Bound Rating AlgoritHm*, [12]) to compute a *least upper delay bounds* (LUDB). A good choice of $\theta^{(h)}$ is computed using linear programs that are optimized to have good performances. In [81], it is proved that the optimal choice can be efficiently computed in the case of *sink-tree* networks (the network topology is a tree and every flow ends at the root of that tree) and that exact performances bounds can be derived. The principle of the computation is to compute the best family of $\theta^{(h)}$ by removing flows of the networks one by one and after the deletion of each flow, computing the residual service curves resulting from it.

As a consequence, it is primary designed for nested tandems networks. However, this approach can be used for modeling general tandem networks, as explained in [82, 83], by cutting the flows to transform the non-nested network into a nested network. An optimization must also be performed to choose the best way of cutting the flows. Unfortunately, in [11], it is shown that even for small nested tandems (an example in given with a source-tree tandem with two servers) the bounds computed are not tight.

Networks satisfying the GPS policy has also received an lot of attention, and initially by Parekh and Gallager in [89, 90]. In these articles, the authors first derive bounds for a single servers and then study networks. They more precisely study policies (the choice of the parameters ϕ_i they know to be stable, even for cyclic networks). More recently, Barta, Nemeth, *et al.* in [10, 88] study more general choices of ϕ and present an algorithm giving a sufficient condition for the stability. This condition is based on the computation of a fixed point of the intermediate arrival curves. If they converge to some finite functions, then the network is stable.

4.4 Tight performance bounds with linear programming

In this section, we present a method that enables to compute the exact worst-case delays in feed-forward networks in the case of arbitrary multiplexing. Moreover, this method may be adapted to other service policies to obtain better bounds than the existing ones. However, in that latter case, the tightness is lost, except for the special case of FIFO networks, that we will mention at the end of the section.

We make the additional assumption that the arrival curves are piece-wise linear affine and concave and that the service curves are piecewise affine and convex. That is, we can write $\alpha_i = \min_p L_p$ and $\beta^{(h)} = \max_p M_p$ where L_p and M_p are affine functions. We present this method in the case of tandem networks, even though it is valid for any feed-forward network and discuss the difficulties that arise for the general case.

4.4.1 Blind multiplexing

Consider one server crossed by one flow. To compute the worst-case delay at time t, we combine those equations:

- strict service curve: $F^{(1)}(t) - F^{(1)}(s) \geq \beta(t-s)$;
- choose $s = start(t)$: $F^{(1)}(s) = F^{(0)}(s)$;
- introduce the arrival date, u of the bit that departs at time t: $s \leq u \leq t$ and $F^{(0)}(u) \geq F^{(1)}(t)$
- arrival curve: $F^{(0)}(u) - F^{(0)}(s) \leq \alpha(u-s)$;

As we chose α piecewise affine concave (*i.e* the minimum of a finite number of affine curves) and β piecewise affine convex (*i.e* the maximum of a finite number of affine curves), all these equalities and inequalities generate linear constraints where u, s, t, $F^{(0)}(t)$... are variables and α and β only are constant. To compute the worst-case delay, it remains to maximize $t-u$ under those constraints.

Computing the worst-case performance in tandem networks can be done by generalizing this linear program backward from server N to server 1.

The linear program

The variables Let us first define the variables of the linear program. Note that, to emphasize the meaning of the variable, we denote them the same way as the date of the function value they represent. Thus, they will mainly be named t_k or $F_i^{(h)}(t_k)$.

- *time variables:* the time constraints are $t_0, \ldots, t_N$ and u with the following interpretation: consider a bit of data that exits the system at time t_N then t_{N-1} is the start of the backlog period of server N at time t_N ($t_{N-1} = start_N(t_N)$) and more generally, $t_{i-1} = start_i(t_i)$. The variable u represents the arrival date in the system of the bit of data considered;

- *functional variables:* the constraints variables are $F_i^{(0)}(t_k)$ and $F_i^{(h)}(t_k)$ for $i \in h$ and $k \in \{h, h-1\}$. Intuitively, $F_i^{(h)}(t_k)$ represents the value of the CAF $F_i^{(h)}$ at time t_k, and the important dates for $F_i^{(h)}$ are t_h, the date at which the bit of interest exits server h and t_{h-1} the start of the backlogged period of server h. The variable $F_i^{(0)}(t_k)$ represents the CAF of flow i, that is an upper bound for $F_i^{(h)}(t_k)$. The variable $F_i^{(0)}(u)$ will also be used to compute the worst-case delay of flow i.

The linear constraints for the worst-case delay of flow i or backlog at node h. Constraints are given inside brackets.

- *time constraints:* $\forall h$, $[\![t_{h-1} \leq t_h]\!]$;
- *service constraints:* $\forall h$, $[\![\sum_{i \in \mathrm{Fl}(h)} F_i^{(h)}(t_h) \geq \sum_{i \in \mathrm{Fl}(h)} F_i^{(h)}(t_{h-1}) + \beta^{(h)}(t_h - t_{h-1})]\!]$
- *start of backlogged period constraints:* $\forall h$, $\forall i \in \mathrm{Fl}(h)$, $[\![F_i^{(h)}(t_{h-1}) = F_i^{(h-1)}(t_{h-1})]\!]$;
- *causality constraints:* $\forall h$, $\forall i \in \mathrm{Fl}(h)$ and $k \in \{h, h-1\}$, $[\![F_i^{(0)}(t_k) \geq F_i^{(h-1)}(t_k) \geq F_i^{(h)}(t_k)]\!]$;
- *non-decreasing constraints:* $\forall i$, $\forall h \in \mu(i)$, $[\![F_i^{(0)}(t_h) \geq F_i^{(0)}(t_{h-1})$; $F_i^{(h)}(t_h) \geq F_i^{(h)}(t_{h-1})]\!]$;
- *arrival constraints:* $\forall i$, $\forall k < h \in \mu(i)$,$[\![F_i^{(0)}(t_h) - F_i^{(0)}(t_k) \leq \alpha_i(t_h - t_k)]\!]$;
- *constraints involving u:* (for the worst-case delay only) $[\![t_{\mu_i(1)-1} \leq u \leq t_{\mathrm{end}(i)}$; $F_i^{(0)}(u) - F_1^{(0)}(t_{\mu_i(1)-1}) \leq \alpha_1(u - t_{\mu_i(1)-1})\}$; $F_i^{(0)}(u) \geq F_i^{(\mathrm{end}(i))}(t_{\mathrm{end}(i)})]\!]$.

The objectives

- for the worst-case delay of flow i:

$$\textbf{Maximize } t_{\mathrm{end}(i)} - u;$$

- for the worst-case backlog at node h:

$$\textbf{Maximize } \sum_{i \in \mathrm{Fl}(h)} F_i^{(0)}(t_h) - \sum_{i \in \mathrm{Fl}(h)} F_i^{(h)}(t_h).$$

For a given network $\mathcal{N}$, let us denote λ the linear program defined above and d_λ its optimal solution if the objective is to find the worst-case delay, and b_λ if the objective is to find the worst-case backlog. The following theorem holds.

Theorem 8. *Let $\mathcal{N}$ be a tandem network with n servers and p flows. The LP instance λ has $\mathcal{O}(pn)$ variables and $\mathcal{O}(pn^2)$ constraints and is such that the optimum is the worst end-to-end delay for flow i is d_λ (resp. the worst backlog at server h is b_λ).*

The proof can be found in [25]. Here, we explain the principle of the proof on an example. It is based on the construction of an admissible trajectory that satisfy the solution of the LP.

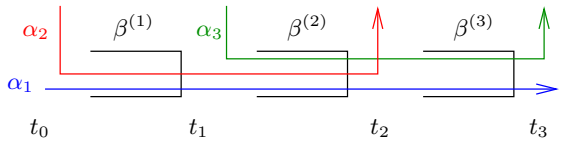

Figure 4.1: Tandem network with 3 nodes and 3 flows.

Consider the network of Figure 4.1. This network has been extensively studied in [99], where it is shown that the exact delay cannot be computed using the application of (min,plus) algebraic methods only. Figure 4.2 explains the relation between the linear program and the trajectories. Given an admissible trajectory, the assignments variables can be read from the figure: let t_3 be exit time of the bit for which the delay is computed. The variables t_i are defined as the beginning of the backlogged

period at each server. Then, the variables $F_i^{(h)}(t_k)$ are assigned the value of the CAF $F_i^{(h)}$ at time t_k. Those variable satisfy of course all the constraints, that are derived from the network calculus basics. Conversely, to reconstruct the trajectories from a solution of the linear program, it is enough to linearly interpolate the CAFs: $F_i^{(\mu_i(1)-1)}$ between $t_{\mu_i(1)-1}$ and $t_{\text{end}(i)}$; and $F_i^{(h)}$ between t_{h-1} and t_h. After this period, the function can be set equal to $F_i^{(\mu_i(1)-1)}$, and before t_{h-1}, equal to $F_i^{(h-1)}$.

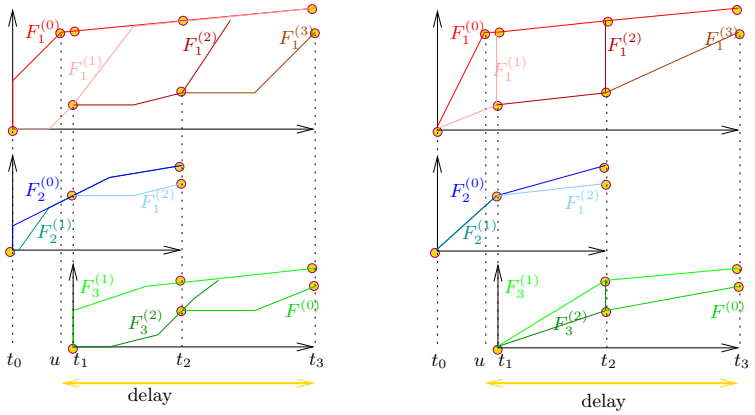

Figure 4.2: From the trajectories to the linear program and back to the trajectories for blind multiplexing.

Leftover service curve Let $\mathcal{N}$ be a network and flow 1 be the flow of interest, until now we have investigated a way to compute the worst delay for fixed constraints $(\alpha_i)_{i\in F}$ and $(\beta_j)_{j\in S}$. One may want to measure how the global network acts upon flow 1, in particular whether some minimum end-to-end service curve β can be guaranteed. It is called a *universal* end-to-end service curve if β is independent of α_1 (*i.e.* β remains an end-to-end service curve for any choice of α_1). Precomputing such an universal curve can be useful to quickly compute a bound on end-to-end delays for flow 1 for several different curves α_1 (thanks to the horizontal distance of Theorem 1).

For tandem networks, it is possible to compute a universal end-to-end service curve which is optimal in the sense that it is maximal for all the universal service curves, using the duality principle of linear programming ([101]).

Generalization to feed-forward networks The same ideas can be applied to general feed-forward networks. But several difficulties arise, making the problem much more difficult to analyze.

- **The number of dates to consider grows exponentially with the size of the network.** Indeed, from one given date used for the start of backlogged period of one given server, several dates must be defined for the beginning of the backlogged period of all the predecessors of this server. As a consequence, the number of dates to define is the number of paths from any node the last server visited by the flow of interest.
- **Those dates are not totally ordered** and every order compatible with the NC constraints must be generated, leading to one different linear program for each different order. Now, the exact worst-case delay is the maximum solution of an exponential number of linear programs.

To illustrate this fact, consider the example of Figure 4.3. The departure date of the bit of interest if $t_\emptyset$, and we can define $t_4 = start_4(t_\emptyset)$, $t_{24} = start_2(t_4)$ and $t_{34} = start_3(t_4)$. But then, for server 1, two start of backlogged period have to be defined: t_{134} and t_{124}. Four orders have to be consider: either t_{24} and t_{34} belong to different backlogged period, in which case we have $t_{124} \leq t_{24} \leq t_{134} \leq t_{34}$ or $t_{134} \leq t_{34} \leq t_{124} \leq t_{24}$, or they belong to the same backlogged period, in which case we have $t_{134} = t_{124} \leq t_{34} \leq t_{24}$ or $t_{134} = t_{124} \leq t_{24} \leq t_{34}$. Date u also has to be inserted in these orders, inducing even more linear programs. The other constraints depending of the order of the dates (arrival, non-decreasing) have to be generated accordingly to these orders.

A reduction of X3C (Exact-three-cover, [65]) to this problem shows that it is in fact NP-hard to compute the exact worst-case performances under the assumptions we made.

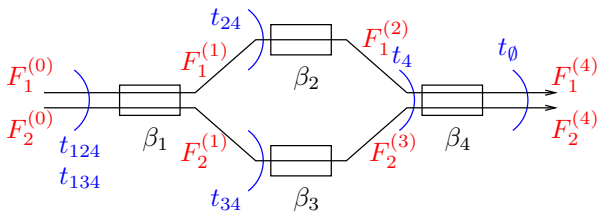

Figure 4.3: Example of a feed-forward network.

Generalization to other service policies The *blind multiplexing*, or *arbitrary multiplexing* gives very pessimistic performance bounds when the service policy is known. It may not always be possible to find a simple linear program that computes the exact worst-case performances efficiently even for tandem networks. For example, an attempt to find a linear program encoding Static Priorities can be found in [26]. As solution to improve the bounds obtained by Algorithm 1 and the exact worst-case for blind multiplexing is to mix those two approaches. This is done by applying the steps of Algorithm 2.

Algorithm 2: Mixing linear program with (min,plus) techniques.

```
1 begin
2 |   Generate the LP program corresponding to the blind multiplexing;
3 |   Compute the intermediate arrival curves for each flow, at each node it crosses using
  |   Algorithm 1;
4 |   Add the linear constraints corresponding to those arrival curves;
5 |   Compute the optimal solution with those constraints.
6 end
```

4.4.2 FIFO multiplexing

Difference with the blind multiplexing case FIFO multiplexing is a special case where it is also possible to compute tight bounds using linear programs. Two elements differ from the blind multiplexing case:

- use the FIFO multiplexing;
- consider simple service curves.

This two elements induce the following linear constraints for one server crossed by two flows: if t is the departure date of the bit of interest,

- *(simple service curve)* there exists $s \leq t$ such that we have $F_1^{(1)}(t) + F_2^{(1)}(t) \geq (F_1^{(0)} + F_2^{(0)}) * \beta(t) = F_1^{(0)}(s) + F_2^{(0)}(s) + \beta(t-s)$, and
- *(FIFO hypothesis)* there exists $u \in [s,t]$ such that $F_1^{(0)}(u) = F_1^{(1)}(t)$ and $F_2^{(0)}(u) = F_2^{(1)}(t)$,

and the monotonicity, causality and arrival curves constraints are still valid. The start of backlog period constraints are dropped. Notice that for one date (t), two new dates have been introduces (s and u). Consequently, the number of dates defined at each server will double and we will also face the problem of the ordering of those variables.

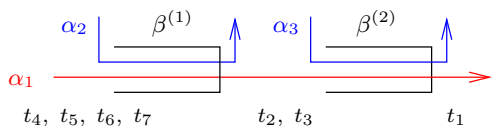

Figure 4.4: Example of a network with two servers and three flows.

Consider the example of two servers and three flows on Figure 4.4. Define t_1, t_2 and t_2 as t, u and s of the previous example. We also write $t_2 = FIFO(t_1)$ and $t_3 = SC(t_1)$. For server 1, we can define $t_4 = FIFO(t_2)$, $t_5 = SC(t_2)$, $t_6 = FIFO(t_3)$ and $t_7 = SC(t_3)$. We know that $t_3 \leq t_2 \leq t_1$, that $t_5 \leq t_4 \leq t_2$, that $t_7 \leq t_6 \leq t_3$ and that $t_7 \leq t_5$ and $t_6 \leq t_4$. But, t_5 and t_6 cannot be ordered. A total order is necessary to get tight bounds to ensure the monotonicity of the functions. This order can be set by the linear program using Boolean variables: consider a sufficiently large constraint M (in our case, this constant is can easily be computed and some LP solver can also compute this constant), b be a Boolean variable (taking value 0 or 1) and the following constraints to force the monotony of f:

$$\begin{aligned} t_5 + bM &\geq t_6 & f(t_5) + bM &\geq f(t_6) \\ t_6 + (1-b)M &\geq t_5 & f(t_6) + (1-b)M &\geq f(t_5) \end{aligned}$$

If $b = 1$, then we have $t_5 + M \geq t_6$, which is a dumb constraint for M large enough, so $t_6 \leq t_5$ and $f(t_6) \leq f(t_5)$; if $b = 0$, we have $t_5 \leq t_6$ and $t_6 + M \geq t_5$ (also dumb for M large enough).

The linear program

The variables More formally, the variables of our problem are the following:

- *time variables:* $t_1, \ldots, t_{2^{N+1}-1}$, where t_{2k} and t_{2k+1} correspond to the FIFO hypothesis and the service curve constraints with regards to t_k, respectively: $t_{2k} = FIFO(t_k)$ and $t_{2k+1} = SC(t_k)$;
- *functional variables:* $F_i^{(h)}(t_k)$ for $h \in [\mu_i(1) - 1, \text{end}(i)]$ and $k \in [2^{n+1-h}, 2^{n+2-h} - 1]$, where we define $F_i^{(\mu_i(1)-1)}$ as $F_i^{(0)}$. If the backlog at server n is computed, then variables $F_i^{(0)}(t_1)$, $i \in \text{Fl}(n)$ must also be defined.

The linear constraints for the delay of flow i or the backlog at node h As said before, the number of dates (hence of variables) grows exponentially with the tandem length, since it doubles at each node as we go backwards and in a multi-node scenario, these dates are only partially ordered. We have (*) $t_{2k+1} \leq t_{2k} \leq t_k$ for $k < 2^n$ and (**) if $t_k \leq t_{k'}$, then $t_{2k} \leq t_{2k'}$ and $t_{2k+1} \leq t_{2k'+1}$ for $2^h \leq k, k' < 2^{h+1}$. The transitive closure of these properties only lead to a partial order of $t_{2^h}, \ldots, t_{2^{h+1}-1}$.

If there exists h such that $2^h \leq k, k' < 2^{h+1}$, t_k and $t_{k'}$ are ordered by introducing a binary variable if necessary. This is done recursively backward the following way:

- For server n, $t_2 \geq t_3$:
- Suppose that dates are ordered for all $k \in [2^{h-1}, 2^h - 1]$ and consider dates t_k, $k \in [2^h, 2^{h+1} - 1]$.
  - First generate the partial *known* order using (*) and (**);
  - if $k = 2\ell$ and $k' = 2\ell'$ or if $k = 2\ell + 1$ and $k' = 2\ell' + 1$, use the variable that orders t_ℓ and $t_{\ell'}$;
  - for any other pair, introduce a new binary variable.

If variables x and y use the variable b to be ordered, we note $x \leq_b y$ to represent the constraints

$$\begin{aligned} x + bM &\geq y \\ y + (1-b)M &\geq x. \end{aligned}$$

Also, if no binary variable is needed, we write $x \leq_\emptyset y$ to represent the constraint $x \leq y$.

We can now write the linear constraints:

- *time and monotonicity constraints:* if $t_k \leq_b t_{k'}$, then $[\![t_k \leq_b t_{k'};\ F_i^{(h)}(t_k) \leq_b F_i^{(h)}(t_{k'})]\!]$;
- *FIFO hypothesis:* if $\mu(1) \leq h$ then $[\![F_i^{(h)}(t_k) = F_i^{(h-1)}(t_{2k})]\!]$;
- *service constraints:* $[\![D^{(h)}(t_k) \geq A^{(h-1)}(t_{2k+1}) + \beta_h(t_k - t_{2k-1})]\!]$;
- *arrival constraints:* if $t_k \leq_\emptyset t_{k'}$ then $[\![F_i^{(0)}(t'_k) - F_i^{(0)}(t_k) \leq \alpha_i(t_{k'} - t_k)]\!]$; if $\exists b \neq \emptyset$ and $t_k \leq_b t_{k'}$ then $[\![F_i^{(0)}(t_{k'}) - F_i^{(0)}(t_k) \leq \alpha_i(t_{k'} - t_k) + (1-b) \cdot M,\ F_i^{(0)}(t_k) - F_i^{(0)}(t_{k'}) \leq \alpha_i(t_k - t_{k'}) + b \cdot M]\!]$;
- *additional arrival constraints for the backlog at server n:* if $i \in \text{Fl}(n)$ then $[\![F_i^{(0)}(t_1) - F_i^{(0)}(t_{2^{n-\mu_i(1)-1}}) \leq \alpha_i(t_1 - t_{2^{n-\mu_i(1)-1}})]\!]$.

Objective If the objective is to compute the worst-case delay, then the objective of the LP is then $\max t_{2^{n-\text{end}(i)}} - t_{2^{n-\mu_i(1)-1}}$.

If the objective is to compute the worst backlog at server n, then the objective of the LP is $\max \sum_{i \in \text{Fl}(n)} F_i^{(\mu_i(1)-1)}(t_1) - \sum_{i \in \text{Fl}(n)} F_i^{(h)}(t_1)$.

For a given network $\mathcal{N}$, let us denote by λ the linear program defined above and d_λ its optimal solution if the objective is to find the worst-case delay, and b_λ if the objective is to find the worst-case backlog. The following theorem holds.

Theorem 9. *The worst-case delay for flow i is d_λ and the worst-case backlog at server n is b_λ*

This can be generalized with no difficulty to any server. The proof of this theorem can be found in [33]. We here only illustrate using the example of Figure 4.3.

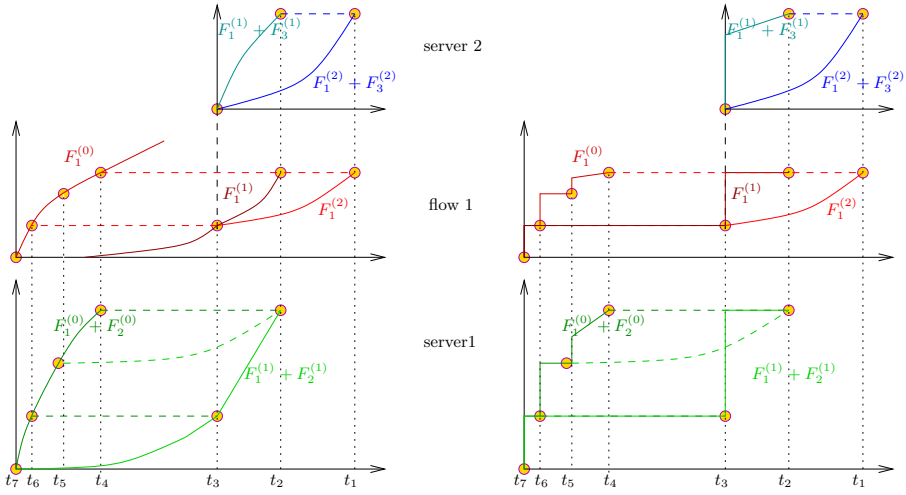

Figure 4.5: From the trajectory to the linear program and back to the trajectories for the network of Figure 4.3. Left: from the trajectory to the linear program: the circles represent the values associates to the variables of the form $F_i^{(h)}(t_k)$. Right: from those variables, it is possible to draw a new trajectory satisfying the constraints where the arrival are maximized according to their constraints, staircase shaped for the intermediate flows, and follows the service curve for the last server.

4.4.3 Numerical experiments

To illustrate the improvements of the performance bounds compared to existing methods, consider the following example of a tandem network composed of 6 servers and one flow crossing all the servers while interfering flows cross two servers at most, as shown on Figure 4.6. We use the following characteristics for the flows and servers: $\forall h \in \{1, \ldots, n\}$, $\beta^{(h)} = \beta : \ t \mapsto R(t-T)_+$ with $R = 10$ and $T = 1$ and $\forall i \in \{1, \ldots, m\}$, $\alpha_i = \alpha : t \mapsto \sigma + \rho t$ where $\sigma = 1$ and $\rho = RU/3$ where $U \in]0, 1[$.

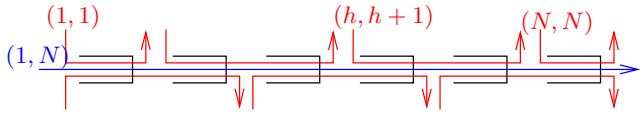

Figure 4.6: Non-nested tandem with 6 servers.

The results are depicted in Figure 4.7, when U varies from 0.3 to 1: for arbitrary multiplexing on the left (compared with the SFA algorithm) and for FIFO multiplexing on the right (compared with the lower and upper bounds obtained with the software Deborah v_{Deb} and V_{Deb}). Note that in the first case, the delay using SFA delay is computed using priorities, as it has been shown in [26] that the Earliest-Deadline-First policy is a policy leading to the worst-case delay of arbitrary multiplexing. In this latter case, we only computed lower and upper bounds (v_{LP} and V_{LP}) of the exact one, using a linear program

that does it more efficiently that the one described here: the upper bound discards about the monotony of the CAFs and CDFs, so that the linear program has no integer variable, and the lower bound forces the equality of some dates so that there is only a quadratic number of dates in the linear program.

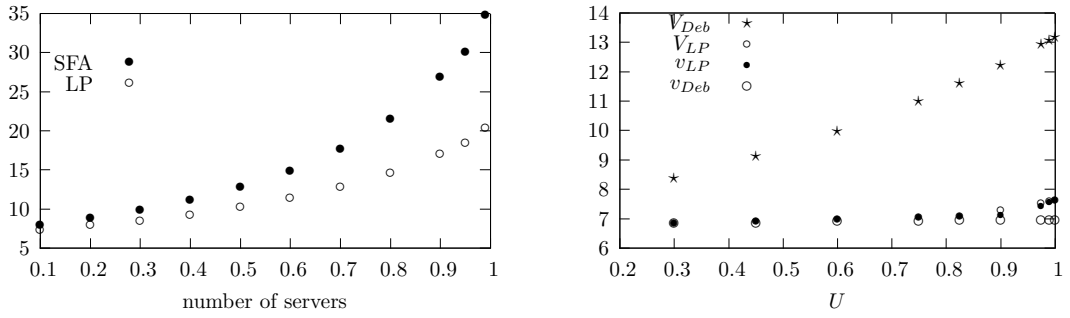

Figure 4.7: Numerical comparisons of the performance bounds.

In this chapter, we presented algorithms to compute tight performance bounds in feed-forward networks, using linear programs. While this method is quite general (it only assumes that the curves are piecewise affine and convex or concave, which is classical, but make it pessimistic for packet flows), it has some drawbacks: first, with the notable exception of tandem networks in blind multiplexing, those algorithm are doubly exponential in time (number of constraints and number of linear programs/integer variables). This issue can be partially overcome by relaxing some constraints in order to obtain a single linear program with no integer variable. Second, it has no simple generalization to non feed-forward and feedback controlled networks. For some service policies, such as static priorities or some GPS policies, it is possible to compute good upper bound by applying the LP techniques on the unfolding of the network, because the network is known to be stable. But how to obtain (possibly) finite delay upper bound when a network is not known to be stable is still an open problem.

Chapter 5

Algorithms for (min,plus) functions

In the previous chapters, we have made an extensive use of the (min,plus) operators to compute performance bounds in networks, and used those operations as basic blocks for our algorithms. It is then an important issue to have efficient implementations of those operations. The question is natural in Network calculus, but also arises in other domains, and in this chapter we will present one such example from the numerical analysis. Section 5.1 is devoted to the description of a large class of functions for which the operations can be computed. Details and proofs can be found in [34]. In Section 5.2, we focus on some special cases of the (min,plus) convolution, that is when concave and convex functions are involved, the convolution can be computed very efficiently. Then, in Section 5.3, we focus on a new application: the computation of long-term weak-KAM integrators. The last two sections are the result of a collaboration with Erwan Faou and Maxime Zavidovique.

5.1 A stable class for the Network calculus operators

Until now, we have mainly dealt with left-continuous, non-negative and non-decreasing functions. For sake of generality, we consider the larger space of functions defined on the non-negative reals or integers, with values within $\overline{\mathbb{R}_{\min}} = \mathbb{R} \cup \{-\infty, +\infty\}$. The Network Calculus makes use of the following operations: minimum, addition, convolution, deconvolution, sub-additive closure, maximum and subtraction.

Depending on whether the functions are defined on $\mathbb{N}$ or $\mathbb{R}_+$ (until now we only dealt with function defined on $\mathbb{R}_+$, we will denote by $\mathcal{D}$ the set of all functions from $\mathbb{N}$ into $\overline{\mathbb{R}_{\min}}$ (*discrete model*) and by $\mathcal{F}$ the set of all functions from $\mathbb{R}_+$ into $\overline{\mathbb{R}_{\min}}$ (*fluid model*). Let $f \in \mathcal{D}$ or $\mathcal{F}$, the subset $Supp(f) = \{t \in X \mid |f(t)| < +\infty\}$ is called the *support* of f. To emphasize this support, we may write $f : Supp(f) \to \mathbb{R}$.

A class of functions is *closed* under some set of operations if combining members of the class with any of these operations outputs (if defined) a function which remains in the class. The *closure* of a class of functions under some set of operations is the smallest class containing these functions and closed under these operations.

Asymptotic behaviors. Let f be a function from X into $\overline{\mathbb{R}_{\min}}$ where $X = \mathbb{N}$ or $\mathbb{R}_+$, then, with $X^* = X \setminus \{0\}$:

- f is *affine* if $\exists \sigma, \rho \in \mathbb{R}, \forall t \in X, f(t) = \rho t + \sigma$ or $\forall t \in X, f(t) = +\infty$ (resp. $-\infty$).
- f is *ultimately affine* if $\exists T \in X$, $\exists \sigma, \rho \in \mathbb{R}$, $\forall t > T, f(t) = \rho t + \sigma$ or $\forall t > T, f(t) = +\infty$ (resp. $-\infty$).
- f is *pseudo-periodic* if $\exists (c, d) \in \mathbb{R} \times X^*$, $\forall t \in X, f(t + d) = f(t) + c$.
- f is *ultimately pseudo-periodic* if $\exists T \in X$, $\exists (c, d) \in \mathbb{R} \times X^*$, $\forall t > T, f(t + d) = f(t) + c$.
- f is *ultimately plain* if $\exists T \in X$, $\forall t > T$, $f(t) \in \mathbb{R}$, or $\forall t > T$, $f(t) = +\infty$, or $\forall t > T$, $f(t) = -\infty$.
- f is *plain* if it is ultimately plain as above, and $\forall\ 0 \leq t < T$, $f(t) \in \mathbb{R}$, and $f(T) \in \mathbb{R}$ or possibly $f(T) = +\infty$ (resp. $-\infty$) in case $\forall t > T$, $f(t) = +\infty$ (resp. $-\infty$).

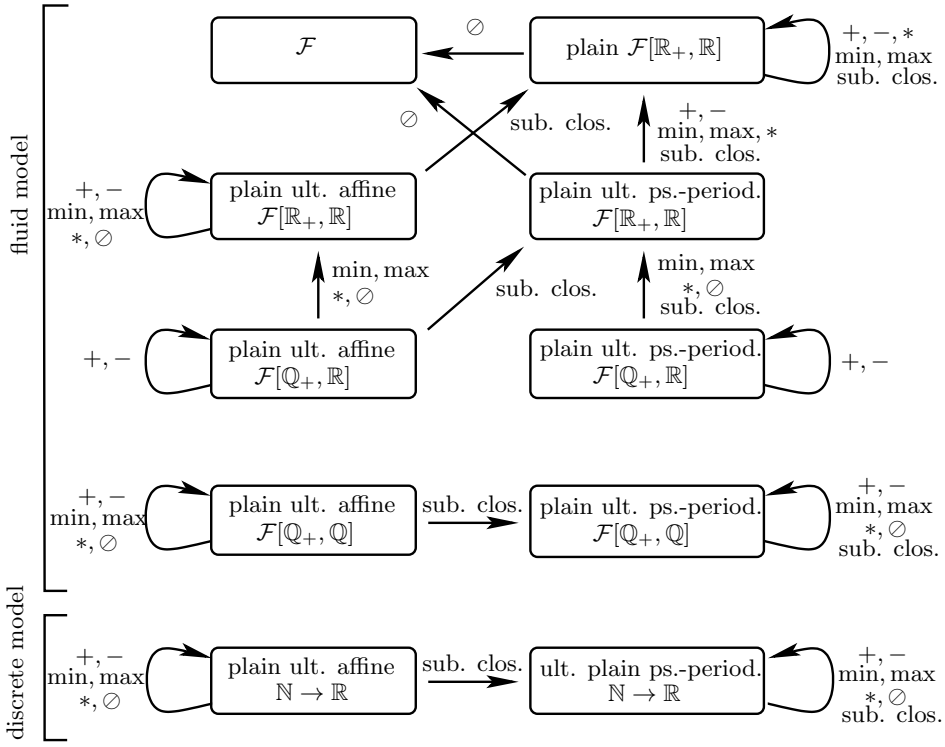

Figure 5.1: Stability/instability of some classes of functions.

For affine and ultimately affine functions, ρ is the *growth rate*. For a pseudo-periodic function f, d is called *a period* of f, c is its associated *increment*, and *the period* of f is its smallest period (if different from 0). For an ultimately affine (resp. ultimately pseudo-periodic) function, we also say that it is ultimately affine (resp. ultimately pseudo-periodic) *from* T, and we say that T is a *rank* of the function. Being plain is equivalent to have a support equal to $[0, T]$ or $[0, T[$ where $T \in \mathbb{R} \cup \{+\infty\}$. A non-decreasing function is always ultimately plain, and if $f(0) \in \mathbb{R}$, it is plain.

Piecewise affine functions. We say that a function $f \in \mathcal{F}$ is *piecewise affine* if there exists an increasing sequence $(a_i)_{i \in \mathbb{N}}$ which tends to $+\infty$, such that $a_0 = 0$ and $\forall i \geq 0$, f is affine on $]a_i, a_{i+1}[$, i.e. $\forall t \in]a_i, a_{i+1}[$, $f(t) = +\infty$ or $\forall t \in]a_i, a_{i+1}[$, $f(t) = -\infty$ or $\exists \sigma_i, \rho_i \in \mathbb{R}$, $\forall t \in]a_i, a_{i+1}[$, $f(t) = \sigma_i + \rho_i t$. The (a_i)'s are called *discontinuities*.

Let $X \subseteq \mathbb{R}_+$ and $Y \subseteq \mathbb{R}$, we denote by $\mathcal{F}[X, Y]$ the set of all piecewise affine functions in $\mathcal{F}$ such that there exists a sequence $(a_i)_{i \in \mathbb{N}}$ with the properties above and satisfying $\forall i \geq 0$, $a_i \in X$ and $f(a_i), f(a_i+), f(a_i-) \in Y \cup \{-\infty, +\infty\}$.

Theorem 10.
- *The class of plain ultimately pseudo-periodic functions of $\mathcal{D}$ is stable under the Network calculus operations, that is $+$, $-$, min, max, $*$, $\oslash$ and the sub-additive closure.*
- *The class of plain ultimately pseudo-periodic functions of $\mathcal{F}[\mathbb{Q}_+, \mathbb{Q}]$ is stable under the Network Calculus operations $+$, $-$, min, max, $*$, $\oslash$ and the sub-additive closure.*

Figure 5.1 summarizes the stability of different classes under those operators. The two classes $\mathcal{D}$ and $\mathcal{F}[\mathbb{Q}_+, \mathbb{Q}]$ are good candidates for the implementation of the algorithms. Indeed, the functions of these classes can easily be stored, by chained lists for example: it suffices to store the segments of the transient and first period part (starting point x_i, value at that point $f(x_i)$, right limit $f(x_i^+)$ and slope ρ_i) plus the period, the increment and the rank from which the periodic behavior starts. The data structure is depicted of Figure 5.2.

Another view of this problem would have been to search the smallest class of functions that are stable under those operations and contain some *basic functions*. The PhD thesis of Laurent Jouhet [73] answers this problem when the basic functions are the affine functions, and the class of functions

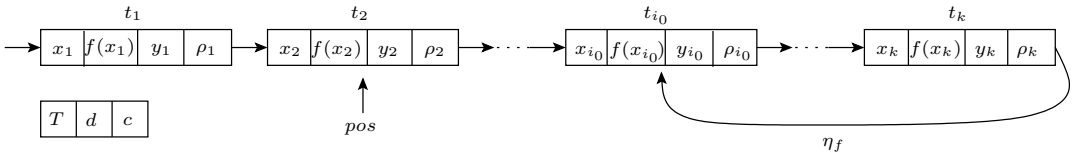

Figure 5.2: A simple data structure to store ultimately pseudo-periodic functions.

that could be reconstructed is the class of ultimately pseudo-periodic piecewise affine functions that possibly have a single discontinuity at 0. Indeed, all the operators preserve the continuity except the sub-additive convolution that can created a discontinuity at 0 (for example, if $f : t \mapsto \sigma + \rho t$ and $\sigma > 0$, $f^* : 0 \mapsto 0;\ t \mapsto \sigma + \rho t$. The functions δ_0 and ε can also be generated. More discontinuities can be introduced by including the class of δ_T functions.

5.1.1 Implementations

Some implementations of Network Calculus have been proposed. Some are very general while some other are valid only for limited classes of functions. Also, they may implement only parts of the network calculus operators, depending on their final aim.

- The DISCO Network Calculator is a Network Calculus Java library aiming at analyzing feed-forward networks [69]. Its principles are detailed in [98]. The algorithms are specially designed for arrival/service curves which are piecewise affine concave and convex functions and performs an SFA (separated flow analysis, similar to Algorithm 1) or TFA (total flow analysis, an even more pessimistic computation) for several service policies.

- The Real-Time Calculus Toolbox (RTC) is a Matlab toolbox for performance analysis of distributed real-time and embedded systems [105, 106]. Its Java kernel implements the main Network Calculus operations, except the sub-additive closure. It deals with piecewise affine functions defined over $\mathbb{R}_+$ which are not necessarily increasing nor positive, but which have a periodic behavior from a point. Infinite values are also allowed. This class is very close to the classes introduced here where this asymptotic behavior is called ultimate pseudo-periodicity. Besides correctness, no complexity analysis has been given as far as we know. They use their algorithms to implement various service policies and deal with networks modeled as event graphs.

- Another software called CyNC is based on Matlab and Simulink, and implements the Network Calculus operations, except the sub-additive closure [95, 96]. It only considers input functions defined over $\mathbb{R}_+$ which are staircases up to a point from which they are affine. It seems that it uses some brute force algorithms, but apparently their correctness and complexity have not been precisely studied.

- The software COINC [15] was the first to implement the algorithms presented in [34], as a stand alone version or a Scilab toolbox.

- The second software based on the algorithms of [34] is PEGASE ([36, 40]). It implements the most general class of functions as well as simpler classes such as concave of convex the have to linear-time algorithms. This software is also intended to compute worst-case bounds for very large networks, under various service policies, such as AFDX [39]. Some optimizations concerning the complexity of stair-case have been implemented, allowing to deal efficiently with packet flows [38].

5.2 Fast convolution by a convex function

In this section we focus on the (min,plus) convolution by a convex function. As we chose to represent the functions as the list of their successive segments, we assume here that the functions are ultimately affine. Proof of the results can be found in [16].

An *affine segment* is a function whose support is an interval and such that the function is affine on this interval. If g is an affine segment, then its slope is denoted by g'.

Let $f : [a, b] \to \mathbb{R} \in \mathcal{F}$ be a piecewise affine function on $\mathbb{R}_+$ with values in $\mathbb{R}_{\min}$. We assume that the support of f is a closed interval, but the result holds for open or semi-open intervals. There exists $a_0 = a < a_i < \cdots < a_n = b$ and $f_i : [a_{i-1}, a_i] \to \mathbb{R}$ affine segments such that $f = \min_{i=1}^{n} f_i$. Then, for $i \in \{1, \ldots, n\}$, f_i' is the slope of f on $[a_i, a_{i+1}]$. Using the distributivity of the convolution over the minimum ([34, 79]), we have

Lemma 1 (convolution of a convex function by an affine function)**.** *Let $f : [a, b] \to \mathbb{R}$ be a convex piecewise affine function and $g : [c, d] \to \mathbb{R}$ be an affine segment of slope g'. Then $f * g : [a+c, b+d] \to \mathbb{R}$ is a convex piecewise affine function defined by*

$$f * g(x) = \begin{cases} f(x-c) + g(c) & \text{if} \quad a + c \leqslant x \leqslant \alpha + c, \\ f(\alpha) + g(x-\alpha) & \text{if} \quad \alpha + c < x \leqslant \alpha + d, \\ f(x-d) + g(d) & \text{if} \quad \alpha + d < x \leqslant b + d, \end{cases}$$

where $\alpha = \min\{a_i \text{ in the decomposition of } f \mid f_i' \geqslant g'\}$.

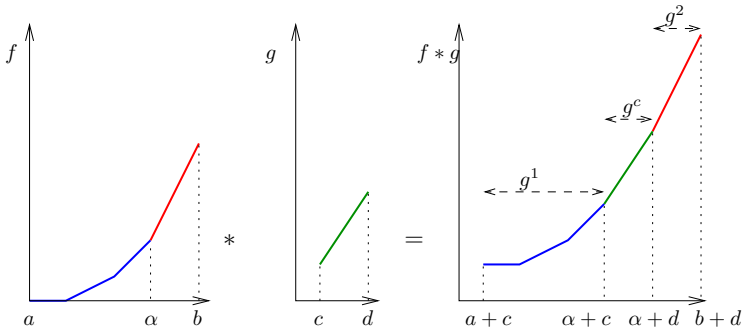

Figure 5.3: Convolution of a convex function by an affine function and decomposition into three functions.

Figure 5.3 illustrates this lemma. In the rest of the section, we will use a decomposition of such a convolution into three parts : $f * g = \min(g^1, g^c, g^2)$, where (i) $g^1 = f * g_{|[c+a, c+\alpha]}$; (ii) $g^c = f * g_{|[c+\alpha, d+\alpha]}$; (iii) $g^2 = f * g_{|[d+\alpha, d+b]}$.

In other words, g^1 is composed of the segments of f whose slope are strictly less than that of g, g^c corresponds to the segment g and g^2 is composed of the segments of f whose slope are greater than or equal to that of g. Note that g^c is also concave.

5.2.1 Convolution of two convex functions

A direct consequence of this lemma is the following theorem, stated in [79]. A complete proof is presented in [30].

Theorem 11 (convolution of a convex function by a convex function)**.** *If f and g are convex and piecewise affine, then $f * g$ is obtained by putting end-to-end the different linear pieces of f and g sorted by increasing slopes.*

For sake of completeness, we give below Algorithm 3 for computing the (min,plus)-convolution of two convex functions defined on $\mathbb{N}$ on a finite support.

5.2.2 Convolution of a concave function by a convex function

Now, consider two functions $f : [a, b] \to \mathbb{R}$, convex, and $g : [c, d] \to \mathbb{R}$, concave on its support, with respective decompositions in $f_i : [a_{i-1}, a_i] \to \mathbb{R}$, $i \in \{1, \ldots, n\}$ and $g_j : [c_{j-1}, c_j] \to \mathbb{R}$, $j \in \{1, \ldots, m\}$. Then $g = \min_{j=1}^{m} g_j$ and $f * g = \min_j^m f * g_j$.

The following lemma, that considers two consecutive affine functions of g, leads to an efficient algorithm to compute the convolution of a convex function by a concave function. It is illustrated by Figure 5.4

Algorithm 3: Convolution of two convex functions

Data: $f : [0, n] \to \mathbb{R}$ a convex function with slopes (r_i), $g : [0, m] \to \mathbb{R}$ a convex function with slopes (ρ_i).
Result: $h = f * g$

```
1 begin
2     i ← 0; j ← 0; h(0) ← f(0) + g(0);
3     while i + j < n + m do
4         if i ≠ n and (r_i < ρ_j or j = m) then
5             h(i + j + 1) ← h(i + j) + r_i; i ← i + 1;
6         else
7             h(i + j + 1) ← h(i + j) + ρ_j; j ← j + 1;
8 end
```

Lemma 2. *Consider the convolutions $f*g_{j-1}$ and $f*g_j$. Let $\alpha_j = \min\{a_i$ in the decomposition of $f \mid f'_i \geq g'_j\}$ and $\alpha_{j-1} = \min\{a_i$ in the decomposition of $f \mid f'_i \geq g'_{j-1}\}$. Then*

- $\forall x \leqslant c_j + \alpha_j,\ f * g_j(x) \geqslant f * g_{j-1}(x)$;
- $\forall x \geqslant c_j + \alpha_{j-1},\ f * g_{j-1}(x) \geqslant f * g_j(x)$.

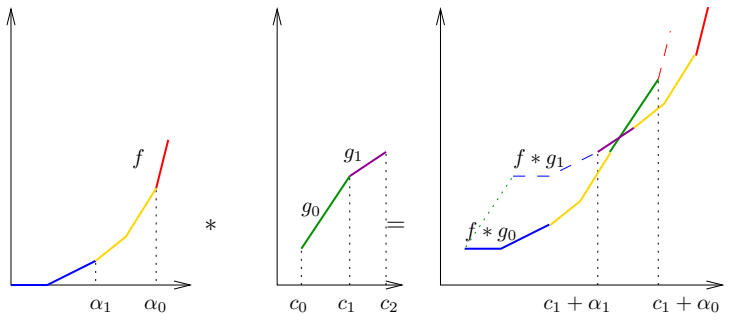

Figure 5.4: Convolution of a convex function by a concave function.

Another formulation of Lemma 2 is that $g_j^1 \geq f * g_{j-1}$, $g_{j-1}^2 \geq f * g_j$, and that the two functions intersect at least once. Hence g_j^1 and g_{j-1}^2 cannot appear in the minimum of $f * g_j$ and $f * g_{j-1}$. By transitivity, there is no need to compute entirely the convolution of the convex function by every affine component of the decomposition of the concave function. If there are more than two segments, successive applications of this lemma show that only the position of the segments of the concave function must be computed, except for the extremal segments. As a consequence, the convolution of a concave function by a convex function can be computed by Algorithm 4.

The following theorem is another consequence of this lemma and is more precise about the shape on the convolution of a convex function by a concave function.

Theorem 12 (convolution of a convex function by a concave function)**.** *The (min,plus)-convolution of a convex function by a concave function can be decomposed in three (possibly trivial) parts: a convex function, a concave function and a convex function.*

If the concave function is composed of m segments and the convex function of n segments, then the convolution of those two functions can be computed in time $\mathcal{O}(n + m \log m)$. The $\log m$ term comes from the fact that one has to compute the minimum of m segments (see [34] for more details). If the functions are now defined on $\mathbb{N}$, then, as no intersection point has to be computed for the minimum, the time complexity is $O(n + m)$. The corresponding algorithm is given in Algorithm 4, where without loss of generality (the (min,plus)-convolution is shift-invariant), the functions f and g are defined on $\mathbb{N}$ and finite between respectively 0 and n, and 0 and m. The slopes of the functions are thus $f'_i = f(i) - f(i-1)$ and $g'_i = g(i) - g(i-1)$.

Algorithm 4: Convolution of a convex function by a concave function

Data: $f : [0, n] \to \mathbb{R}$ a convex function with slopes (r_i), $g : [0, m] \to \mathbb{R}$ a concave function with slopes (ρ_i).
Result: $h = f * g$

begin
  `/* Initialization */`
  $k \leftarrow 0$;
  while $k \leqslant m + n$ **do** $h(k) \leftarrow +\infty$; $k \leftarrow k + 1$;
  $i \leftarrow 0$; $j \leftarrow 0$; $h(0) \leftarrow f(0) + g(0)$;
  `/* First convex part of the convolution */`
  while $f'_i \leqslant g'_0$ **do**
    $i \leftarrow i + 1$; $h(i) \leftarrow f(i) + g(0)$;
  `/* Concave part of the convolution */`
  $j \leftarrow j + 1$; $h(i + j) \leftarrow f(i) + g(j)$;
  while $j < m$ **do**
    while $g'_j < f'_{i-1}$ **do** $i \leftarrow i - 1$;
    $h(i + j) \leftarrow \min(h(i + j), f(i) + g(j))$;
    $h(i + j + 1) \leftarrow \min(h(i + j + 1), f(i) + g(j + 1))$;
    $j \leftarrow j + 1$;
  `/* Second convex part of the convolution */`
  while $i < n$ **do**
    $i \leftarrow i + 1$; $h(i + m) \leftarrow \min(h(i + m), f(i) + g(m))$;
end

Note that the shape of the function is simplified when the functions have infinite supports ($b = +\infty$ or $d = +\infty$). Indeed, either the last slope of f is less that the last slope of g, in which case each slope of f is smaller that each slope of g and $f * g = f + g(0)$ (assuming without loss of generality that $a = c = 0$). Or the last slope of g is less than the last slope of f, in which case $f * g$ is composed of a convex function and a concave function only, as the convolution of f by the last segment of g is ultimately affine with ultimate rate the slope of the last segment of g. Then the final convex part does not appear.

5.3 Application to fast weak-KAM integrators

Consider the Hamilton-Jacobi equation

$$\partial_t u + H(t, x, \nabla u) = 0, \quad u(0, x) = u_0(x), \tag{5.1}$$

where $H(t, x, v)$ is a Hamiltonian function, periodic in t and x

$$H : \mathbb{R}_+ \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \tag{5.2}$$

and where u_0 is a Lipschitz function. We will consider the case where H is separable and convex in p: $H(t, x, p) = \frac{1}{2}|p + P|^2 + V(t, x)$, with V a $\mathcal{C}^2(\mathbb{R}_+ \times \mathbb{R}^n)$ and bounded function and $P \in \mathbb{R}^n$. The solution of this equation can be written as

$$u(t, x) = \inf_{\gamma(t)=x} u_0(\gamma(0)) + \int_0^t L(s, \gamma(s), \dot{\gamma}(s))\mathrm{d}s, \tag{5.3}$$

where the infimum is taken over all absolutely continuous curves $\gamma : [0, t] \to \mathbb{R}^n$ such that $\gamma(t) = x$, and where $L(t, x, v)$ is the Lagrangian associated with H:

$$L(t, x, v) = \sup_{p \in \mathbb{R}^n} \Big(p \cdot v - H(t, x, p) \Big).$$

In our case, $L(t, x, v) = \frac{1}{2}|v|^2 - P \cdot v - V(t, x)$.

For periodic Hamiltonians (H is 1-periodic in both t and x), the weak-KAM theorem (see [58, 50]) shows the existence of a constant $\overline{H}$ such that

$$\frac{1}{t}u(t,x) \to \overline{H} \quad \text{when} \quad t \to +\infty.$$

An efficient implementation of a numerical scheme for the Hamilton-Jacobi equation that respects the long time behavior of the exact solution $u(x,t)$ enables to approximate $\overline{H}$.

Using the explicit expression of L, we can approximate the scheme with the following dynamic on the discrete grid $G_\varepsilon = \varepsilon\mathbb{Z}^n$ and for a time step τ:

$$\forall x \in G_\varepsilon, \quad \mathcal{T}^\tau_{t,\varepsilon}u(x) = \inf_{y\in G_\varepsilon}\left(u(y) + \frac{\tau}{2}\left|\frac{x-y}{\tau}\right|^2 - P\cdot(x-y)\right) - \tau V(t,x). \tag{5.4}$$

involving a (min,plus)-convolution of u with a convex function.

For a given integer N, we define

$$\mathcal{T}^{N\tau}_{t,\varepsilon}u = \mathcal{T}^\tau_{t_{N-1},\varepsilon} \circ \cdots \circ \mathcal{T}^\tau_{t_1,\varepsilon} \circ \mathcal{T}^\tau_{t_0,\varepsilon}u$$

where $t_i = t + i\tau$.

In case where $V(t,x)$ is periodic, using the weak-KAM theory:

Theorem 13. *For any $\varepsilon = \frac{1}{k}$ and $\tau = \frac{1}{\ell}$ with $k,\ell \in \mathbb{N}$, there exists a unique constants $\overline{\mathcal{H}}_{\varepsilon,\tau}$ such that if u is any bounded initial datum on G_ε at $t = 0$, then we have in L^∞*

$$\frac{1}{N\tau}\mathcal{T}^{N\tau}_{0,\varepsilon}u \longrightarrow \overline{\mathcal{H}}_{\varepsilon,\tau}, \text{ as } N \to +\infty.$$

if moreover ε, τ and ε/τ are small enough, then there exists a constant M such that

$$\left|\overline{\mathcal{H}}_{\varepsilon,\tau} - \overline{H}\right| \leqslant M\left(\frac{\varepsilon}{\tau} + \tau\right).$$

Numerical simulations

Now the computation of $\mathcal{T}^\tau_{t,\varepsilon}u(t,x)$ given in (5.4) is made of two steps:

- (min,plus)-convolution of u and $h : x \mapsto \tau K^*(\frac{x}{\tau})$;
- subtract $\tau V(t,x)$.

Note that the (min,plus)-convolution described above is here defined on functions with an unbounded support. But in our case, functions are 1-periodic in x and t, and H is convex in p with a global minimum. Then, it is enough to compute the convolution on a single period only (the (min,plus) convolution preserves the periodicity), and replace h by its restriction on a support of size 2 centered on its minimum. If $\varepsilon = 1/k$ with the notation of the previous section, then both functions u and h are defined on grids of size k and $2k$ respectively.

The convolution of u and h can be efficiently computed following these steps:

1. Decompose u into convex and concave parts. This can be done in linear time: the three first points determine if a part is concave or convex. Then, this part is extended as much as possible while preserving the concavity or convexity and so on.
2. For each convex or concave part, perform the convolution with h using Algorithms 3 or 4.
3. Take the minimum of all these convolutions.

The complexity of this Algorithm is then $\mathcal{O}(ck)$, where c is the number of components in the decomposition of u into concave/convex parts.

Implementation issues The main issue with this algorithm is that c - the number of components in the decomposition of u - can become very large, and then lead to a quadratic time complexity, which is the complexity of a naive algorithm for computing the convolution. Experimentally, we indeed obtained decomposition of large size. It is mainly due to the discretization of u: nearly affine parts, after performing the convolution several times, are fast alternations of convex and concave parts. One solution to make the computation more efficient is to consider that those parts are convex and use Algorithm 3.

This is done by decomposing u into convex and concave parts with a tolerance parameter; we do not request for convex parts to have increasing increments, but the increments to have an increase more than $-\eta$. The choice of an optimal tolerance η, as well as the comparison with parallel implementations, will be the subject of further studies.

Comparison with WENO5 We take the Hamiltonian (5.2) with $P = 1$ and $V(t,x) = 1 - \cos(x)$ on $[-\pi, \pi]$, $k = 600$ grid points, that is $\varepsilon = 1.7e-3$, and $\tau = \sqrt{\varepsilon} = 0.04$. We use the approximated convolution with a tolerance $\eta = 10 \times \varepsilon^2 = 2.7e-5$. The comparisons are made with the fifth-order WENO algorithm (WENO5 see [71]) with 100 grid points, which will be considered here as the exact solution.

We calculate the solution at times $t = 1, 2, 5$ and 15 with initial value $u(0,x) = \cos(2x)$. We see the very good agreement between the solution given by the WENO5 algorithm. The plot, rescaled such that $u(t,-\pi) = 0$, is shown in Figure 5.5. Note that after the time $t = 20$, the solution has converged and remains constant for larger times. The solution observed is thus very close to the weak-KAM solution $u^*(x)$. The number of convex/concave components c is always of order 10 (and equal to 3 - as expected from the shape of the solution - when the stationary state is attained), and the CPU time about 4 times less than the WENO5 algorithm, without any significant deterioration of the accuracy. We show the result in Figure 5.5.

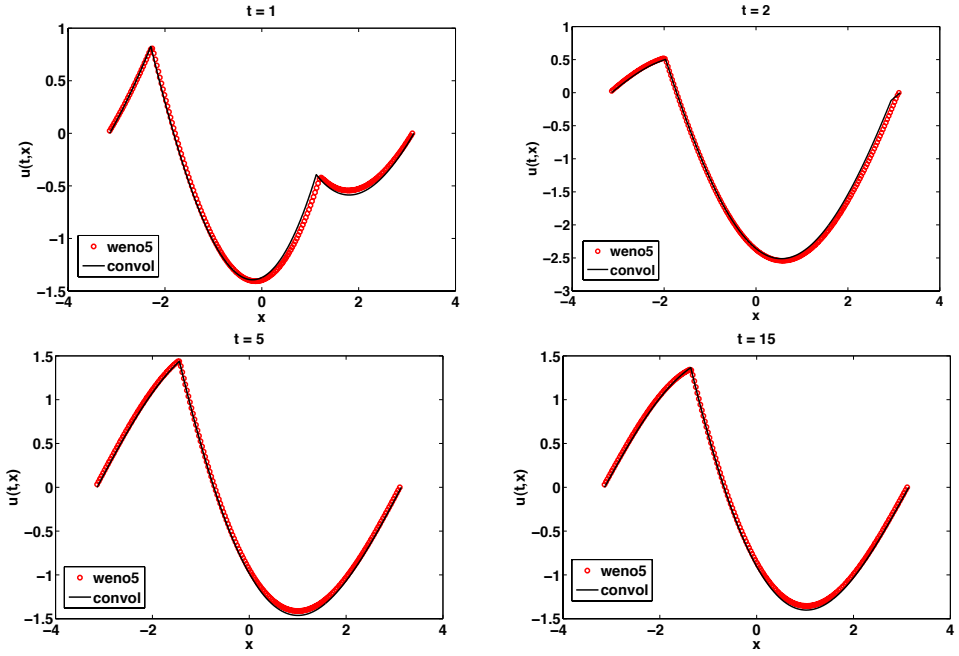

Figure 5.5: Solutions with $\eta = 1.7e-2$. CPU $= 0.3s$ at $t = 15$

We presented in this chapter an application of the (min,plus) convolution algorithm to another domain than network calculus. Note that this is not the first time that tropical algebras are used to study such problems. For example, in [2], Akian et al. use finite (max,plus) elements to model problems of optimal control. Here the approach is different, as the discretization is done in the classical algebra and the (min,plus) framework is used only in the algorithmic part.

Chapter 6

Adapting the constraints to a flow

The concept of arrival curves can be adapted to supervise the behavior of a flow. In that case, the aim is to find an arrival curve that fits the flow. Usually network calculus arrival curves are given as a characteristic of the flow, or a target constraint for the flow that will then cross a shaper so that it has the desired constraint ([45, 79]). The problem of finding the arrival constraint for a CAF has received less attention. It is well-known that given an arrival process A, the best arrival curve α for A is $\alpha = A \oslash A$. If time is discretized, it can also be computed on-line the following way: let $\alpha^{(n)}$ be the best arrival curve for the truncated arrival process $A(0), \ldots, A(n)$. Then $\alpha^{(n+1)}$ can be computed from $\alpha^{(n)}$ in $O(n)$. Indeed, $\alpha^{(n+1)}(n+1) = A(n+1) - A(0)$ and for all $j \leq n$, $\alpha^{(n+1)}(j) = \max[A(n+1) - A(n+1-j), \alpha^{(n)}(j)]$. This formula is not satisfactory. First, the computation might become too costly as n grows. Second, this formula might not be relevant: if the flow's behavior changes, then the arrival curve would not reflect the new behavior and would not be a good representative for the flow.

In this chapter we slightly modify the notion of flow and arrival curve, so they are define to *follow* the behavior of a flow. This has been developed during a collaboration with Alcatel-Lucent in the context supervision of routing protocols. Those protocols are based on the emission of periodic messages that maintain the connectivity and the good behavior of the network. Variations of the small emission rate of the flows might not impact immediately the performance of the network, but might have effect at a longer term. Detecting these variations before a failure occurs is then an important issue.

First flows and constraints are defined in this new context in Section 6.1. The algorithm of detection of the variations of a flow and a variant is presented in Section 6.2 before presenting some numerical results in Section 6.3. This work is part of the PhD of Aurore Junier and is part of a collaboration with Benoît Ronot, from Alcatel-Lucent. More details can be found in [27] and complete proofs in [28].

6.1 Constraints of a flow

Here a flow is a non-decreasing sequence $(x_n)_{n\in\mathbb{N}}$, where $x_0 = 0$ by convention and where x_n is the arrival date of the n-th message of the flow. We assume that $\lim_{n\to\infty} x_n = \infty$.

Graphically, this flow can be represented by the graph $(P_n)_{n\in\mathbb{N}} \in (\mathbb{R}_+ \times \mathbb{N})^{\mathbb{N}}$ where $\forall n \in \mathbb{N}$, $P_n = (x_n, n)$. This graph represents the cumulative number of incoming messages. An example of such a graph is represented on Figure 6.1, where the data flow is $(0, 10, 20, 30, 50, 70, \ldots)$ and the corresponding graph is $((0,0), (10,1), (20,2), (30,3), (50,4), (70,5), \ldots)$.

We slightly modify the notion of arrival curve and generalize it to finite intervals.

Definition 5. *Let $\underline{\alpha}, \overline{\alpha} : \mathbb{R}_+ \to \mathbb{R}_+$ be two non-decreasing functions, $m < n$ be two non-negative integers. The flow (x_n) is* $(\underline{\alpha}, \overline{\alpha})$-constrained *on the interval $[m, n]$ if $\forall m \leq m' \leq n' \leq n$,*

$$\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m' \leq \overline{\alpha}(x_{n'} - x_{m'}).$$

The flow (x_n) is $(\underline{\alpha}, \overline{\alpha})$-constrained *if it is $(\underline{\alpha}, \overline{\alpha})$-constrained on the interval $[0, +\infty[$.*

Graphically, a flow $(x_n)_{n\in\mathbb{N}}$ is $\underline{\alpha}$-lower constrained (resp. $\overline{\alpha}$-upper constrained) if $\forall n \in \mathbb{N}$, $\forall m > n$, P_m is above $\underline{\alpha}$ (resp. below $\overline{\alpha}$) drawn from P_n (respectively denoted as $P_n + \underline{\alpha}$ and $P_n + \overline{\alpha}$).

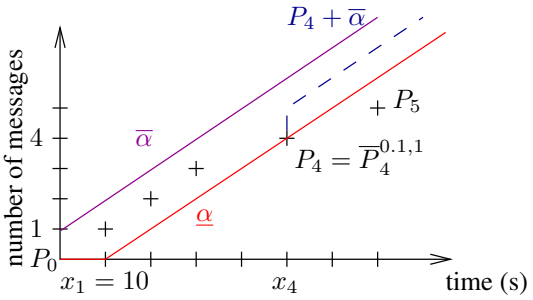

Figure 6.1: Data flow and constraints.

Example 1. *On Figure 6.1,* $\overline{\alpha} : t \mapsto 1 + 0.1t$ *and* $\underline{\alpha} : t \mapsto 0.1(t-10)_+$ *are depicted, as well as* $P_4 + \overline{\alpha}$*. It should be clear that the flow is upper-constrained by* $\overline{\alpha}$*, but not lower-constrained by* $\underline{\alpha}$*. Indeed,* P_5 *is below* $\underline{\alpha}$*, and we have* $5 - 0 = 5 \leq \underline{\alpha}(x_5 - x_0) = 6$*. To conclude, the flow is* $(\underline{\alpha}, \overline{\alpha})$*-constrained on* $[0, 4]$ *but not on* $[0, 5]$*.*

We aim at finding constraints for the arrival flows on long intervals, on the fly. Given a data flow (x_n) and arbitrary curves $\underline{\alpha}$ and $\overline{\alpha}$ such that (x_n) is $(\underline{\alpha}, \overline{\alpha})$-constrained on $[0, n]$, checking that the $n+1$-th messages satisfies those constraints requires $O(n)$ operations and $O(n)$ space. However, when $\underline{\alpha}$ and $\overline{\alpha}$ are respectively chosen affine and rate-latency, this complexity can be drastically reduced to a constant number of operations and constant space. We will use the following notations: $\overline{\alpha}_{\rho,\sigma} : t \mapsto \sigma + \rho t$ and $\underline{\alpha}_{\rho,T} : t \mapsto \rho(t-T)_+$.

Unless otherwise stated, we will assume that σ and T are fixed, and our goal is to guess ρ such that if the flow $(x_n)_{n\in\mathbb{N}}$ is very regular, it is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained. If the traffic is not regular, our goal is to find the successive arrival rates of the messages. In the former case, such a rate ρ exists and is unique.

In the latter case, if we are given the first messages, one needs to check, for the next one, if it satisfies the current constraints and, should the case arise, to compute a new rate. Due to the shape of the functions, this can be done in constant time. Indeed, to check whether a new message is constrained by the current arrival curves, it is enough to check the arrival curves inequality from one message for each curve. We call such a message the (lower and upper) *critical message*, which is the one that will lead to the strongest constraint.

Suppose that from message m, the current constraints are $\underline{\alpha}$ and $\overline{\alpha}$ and the current slope is ρ. The first message that breaks the current constraints is called the *first outgoing message* from n regarding $\underline{\alpha}$ and $\overline{\alpha}$ and defined by

$$o = \min\{p \geq m \mid (x_p) \text{ is not } (\underline{\alpha}, \overline{\alpha})\text{-constrained on } [m, p]\}.$$

The lower critical message can be defined by induction for $n \geq m$ by:

$$\underline{c}(n) = \begin{cases} m & \text{if } n = m \\ n & \text{if } \frac{n}{\rho} - x_n > \frac{\underline{c}(n-1)}{\rho} - x_{\underline{c}(n-1)} \\ \underline{c}(n-1) & \text{otherwise.} \end{cases}$$

The same formula stands for the upper critical message $\overline{c}(n)$, replacing $>$ by $<$.

Figure 6.1 gives an illustration of that: $P_0 + \overline{\alpha}$ is above $P_4 + \overline{\alpha}$, so P_4 gives a strongest constraint than P_0, hence will be called (upper)-critical message. Message 5 is the first outgoing message as it breaks the lower-constraint.

6.2 Computation of the long-term behavior of a flow

We are now ready to present the algorithm that computes the successive arrival curves for a flow. Algorithm 5 gives the elementary functions to detect the outgoing message and update the critical messages using the formulas above.

Algorithm 6 is the algorithm that outputs a list of rates.

Algorithm 5: Elementary functions

```
1 Update_Critical(ρ, c̄, c̲, n)
2 if n < ρ(x_n − x_c̄) + c̄ then c̄ ← n else if n > ρ(x_n − x_c̲) + c̲ then c̲ ← n;
3 Is_Lower_Constrained(T, ρ, n, c̲)
4 if n ≥ ρ(x_n − x_c̲ − T) + c̲ then True else False
5 Is_Upper_Constrained(σ, ρ, n, c̄)
6 if n ≤ ρ(x_n − x_c̄) + σ + c̄ then True else False
```

Algorithm 6: Rates computation

Data: T, σ, (x_n), ρ_0.
Result: list of the rates computed *Rates*.

```
1  begin
2      n ← 0; c̄ ← 0; c̲ ← 0; ρ ← ρ_0;
3      while true do
4          n ← n + 1;
5          if not Is_Lower_Constrained(T, ρ, n, c̲) then
6              ρ ← (n − c̄)/(x_n − x_c̄); c̲ ← n; c̄ ← n ;
7              Rates← Rates::ρ;
8          else if not Is_Upper_Constrained(σ, ρ, n, c̄) then
9              ρ ← (n − c̲)/(x_n − x_c̲) ; c̲ ← n; c̄ ← n;
10             Rates← Rates::ρ;
11         else
12             Update_Critical(ρ, c̄, c̲, n);
13 end
```

If a flow is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained and ρ has been computed by the algorithm, the constraints will always be satisfied for all the next messages and no new rate will be computed. We say that the algorithm has converged in finite time. The next paragraph is devoted to study a special case where this algorithm may converge in finite time, the class of periodic flows..

Periodic flows A flow $(x_n)_{n\in\mathbb{N}}$ is said *N-periodic* if $\forall n \in \mathbb{N}$, $x_{n+N} - x_{n+1+N} = x_n - x_{n+1}$.

Proposition 4. *Let $T, \sigma, \rho \in \mathbb{R}_+$. If (x_n) is a N-periodic, $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained flow, then Algorithm 6 with input $(T, \sigma, (x_n))$ either finds a rate ρ in finite time (and the rate will not be updated anymore) or ultimately has a periodical behavior.*

One could expect that Algorithm 6 converges after a finite time to the arrival rate of a periodic flow. Unfortunately, it is not the case, and increasing σ and T does not help much, as illustrated in Example 2.

Example 2. *Let us consider a 9-periodic flow $x = (0, 15, 35, 55, 80, 100, 120, 140, 160, 180, \ldots)$. This flow is constrained with $\rho = 0.05$, $\sigma = 1$ and $T = 10$. Algorithm 6 alternatively finds $\rho_1 = 0.067$ and $\rho_2 = 0.04$. This computation is represented in Figure 6.2 (left): the first computed rate is $\rho_1 = 1/x_1$. Then, message 4, arriving at x_{o_1} is the first outgoing message and its upper critical message is message 3 arriving at x_{c_1}. The new rate is then $1/(x_4 - x_3) = 0.04$. The next outgoing message is message 10 and its lower critical message is message 9. The next computed rate is the same as the initial rate (one period shift).*

Figure 6.2 (right) shows the behavior of the algorithm when T increases. For example, if $T = 20$ (plain curve), then the first outgoing message is message $o^{20} = 5$ and its upper critical message is $c^{20} = 4$. The new constraints are with $\rho = 0.05$ and the algorithm has converged in finite time. But if $T = 60$ (dashed curves), the behavior of the algorithm will again be periodical: the first outgoing message is message $o^{60} = 13$ and its critical message is message $c^{60} = 12$. The computed rate is still $\rho_2 = 0.04$ and the behavior of the algorithm is still periodic.

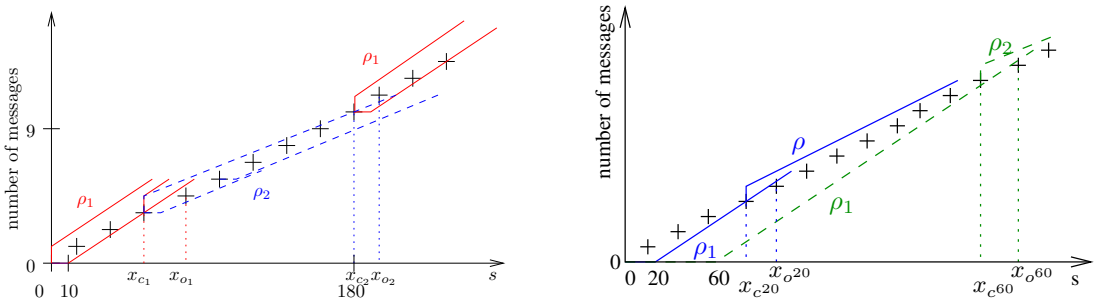

Figure 6.2: Example of rates computed with Algorithm 6 (left) and effect on increasing T (right).

Nevertheless, when the rate computed at some point is close enough to the arrival rates of the messages, Algorithm 6 can converge in finite time.

Theorem 14 (Convergence)**.** *Let (x_n) be a N-periodic, $(\underline{\alpha}_{\rho,T'}, \overline{\alpha}_{\rho,\sigma'})$-constrained flow. There exists ε such that if Algorithm 6 with inputs (x_n), $\sigma > \sigma'$ and $T > T'$ can compute $r \in [\rho - \varepsilon, \rho + \varepsilon]$, it converges to rate ρ in finite time.*

For more regular flows, like balanced flows (*i.e.* such that $\forall n \in \mathbb{N}$, $x_n = \lceil \frac{n}{\rho} \rceil$), one can show (see [27]) that Algorithm 6 converges.

The multi-layered algorithm Algorithm 6 can be made multi-layered. The aim of this is to make the algorithm detect periodic or regular behaviors better and detect only the drastic changes in the flow. The idea is the following.

Layer 0 Perform Algorithm 6 as before but output the sub-flow of the outgoing messages: a new flow is then obtained made of the messages $(\phi_1(i), x_{\phi_1(i)})$ where $\phi_1 : \mathbb{N} \setminus \{0\} \to \mathbb{N} \setminus \{0\}$ is an increasing function where $\phi_1(i)$ is the i-th outgoing message.

Layer 1 Perform Algorithm 6 on the sub-flow $(\phi_1(i), x_{\phi_1(i)})$ and outputs a sub-flow $(\phi_2(i), x_{\phi_2(i)})$ for Layer 2.

Layer ℓ Perform Algorithm 6 on the sub-flow $(\phi_\ell(i), x_{\phi_\ell(i)})$ and outputs a sub-flow $(\phi_{\ell+1}(i), x_{\phi_{\ell+1}(i)})$ for Layer $i+1$.

Example 3. *In the previous example with $T = 10$ and $\sigma = 1$, the first messages of the flow are $(4, 55)$, $(10, 195)$... Figure 6.3 shows that the algorithms then converges on Layer 1 and finds the arrival rate 0.005.*

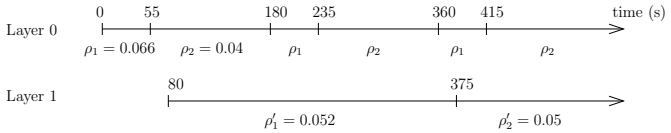

Figure 6.3: Example of Algorithm 6 used with two layers.

It is not known actually if for any periodic flow there exists a layer for which the algorithm converges. Nevertheless, we conjecture it: for any $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained periodic flow, there exists a layer for which the algorithm with parameters $\sigma' > \sigma$ and $T' > T$ converges in finite time. Nevertheless, we also conjecture that for any $\ell \in \mathbb{N} \setminus \{0\}$, there exists a periodic flow such that the algorithm does not converges on the ℓ first layers.

Implementation choices

- *The role of σ and T:* in the previous paragraph, we saw that σ and T do not play an important role: they have no obvious property that make the convergence easier if they are increased (see Example 2). However, those parameters have to be carefully chosen by the user in order to define the tolerance to detect the rate variations. Small parameters will allow a very refined detection. Finally, the choice of these parameters will be made in accordance with the theoretical characteristics of the flow of interest.

- *Rates update:* in the core algorithm, we chose to update the rate when the lower constraint is broken with the upper critical message, and conversely. Another solution would have been to choose the other way round: update the rate using the lower critical message when the lower constraint is broken. Doing this, there is no way to ensure the convergence in finite time.

6.3 Application to the supervision of a flow

6.3.1 Flow slowing down

We first apply the algorithm to a randomly generated flow of messages arriving slower and slower. initially, messages have inter-arrival times uniformly distributed between 1.6 and 2.4 s, and the traffic progressively slows down so that in the end, messages have an inter-arrival uniformly distributed on the interval $[16, 24]$. This simulation could characterize the case where a router is overloaded and then slowly communicates with its neighbors. Figure 6.4 (left) represents such a data flow on the left and the rates computed by the multi-layered version of Algorithm 6 respectively for Layer 0 (center) and Layer 3 (right).

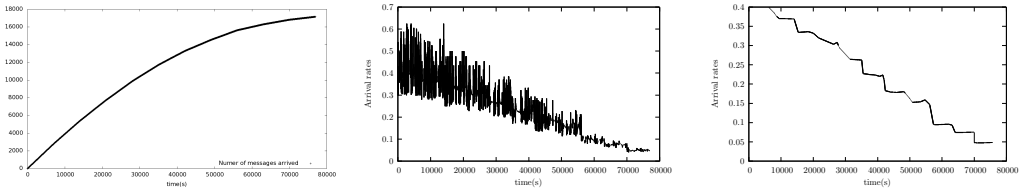

Figure 6.4: Flow of messages arriving slower and slower.

In both cases one can observe the messages slowing down. Due to the fact that the fluctuations are quite important (40%), the rates computed on Layer 0 change frequently. But, on Layer 3, the noise induced by the fluctuations is erased and only the global behavior is observed. This illustrates the fact that the Layer 0 shows many details that are discarded by the next layers.

6.3.2 Flow of a routing protocol

Routing protocols (OSPF - *Open Shortest-paths-first*, [70] - for example) have been defined to route packets efficiently in the network and avoid congestion. The role of these routing protocols is to supervise the network and specially maintaining the connectivity and short paths between each of its components. One classical solution for preserving those properties is that each router sends periodical messages to its neighbors and exchange information with them. This way it is possible for each router or component to maintain the knowledge about the topology of the network. The protocols are usually described in such a way that small deviations from the ideal behavior will not be detected in order to avoid an overload of useless warning messages. In most of the cases, the network will be back to its normal behavior naturally. However, in some cases, the cause of the small deviations is deeper and might result on a major fault without any prior notice.

We also applied the algorithms presented in the previous section to flows generated by an emulated real-size network. The flow is composed of the messages that are used by the OSPF protocol to exchange information about the network (namely the *Hello* and *Link State Advertisement* (LSA) messages) under different settings. Figure 6.5 shows the behavior of this flow when a router periodically reboots.

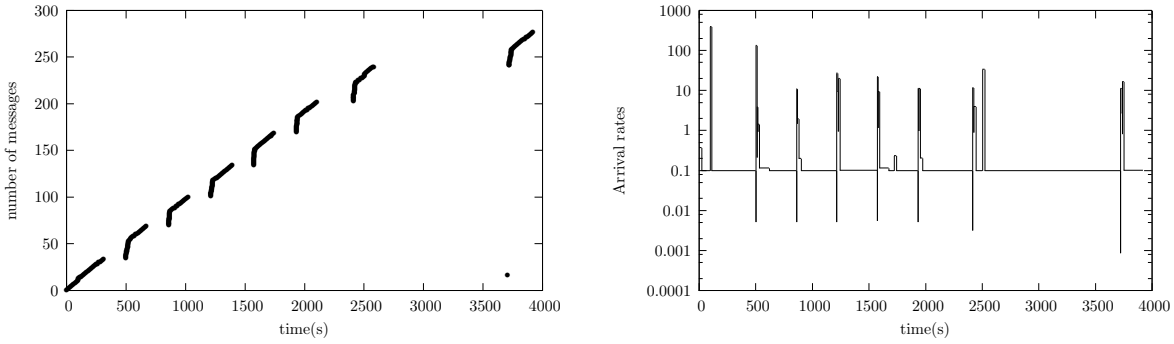

Figure 6.5: Data flow between two routers (left) and the rates computed by Algorithm 6 (right).

In this chapter, we presented an approach using the concept of service curve into a new context: the supervision of a flow. Here, the constraints are not fixed but follow the behavior of a flow, and we gave a first algorithm to find those constraints. Future work include the generalization of this algorithm to handle several flow and search correlation between them. There also might be a link with the multimode network calculus of [91] for analyzing systems whose behavior is periodically changing.

Chapter 7

Conclusion and perspectives

This document presents the main results I recently obtained in the field of network calculus. First, the notion of service curves was investigated and a precise picture of their relationship to each other has been presented, in this document for strict and simple service curves [24]. This shows that there is an intrinsic difficulty in Network residing in the fact that most of the time several models of service curve have to be taken into account.

We also presented a new model of curves, the *packet curve*, that attempts to enable a more precise study of packet flows, whereas most of the study in the manuscript concerns the fluid model. The target application of this model of curves was to study networks that contain flows with packets of different sizes (one packet length could be hundred times the length of other packets), such as wormhole networks. In those kinds of networks, the buffers have a size less than the size of a packet and consequently a packet can occupy several buffers simultaneously and block other packets. Accurate delay bounds could be obtained for toy networks, but, unfortunately, we did not succeed to apply this technique to real-size networks and get better results than the ones obtained with classical network calculus techniques in [59, 94]. This question is still a challenge. Nevertheless, for packet service policies (such as round-robin), we obtained better performance guarantees than with the classical model taking into account only the smallest and largest packet size.

Second, we presented techniques to compute exact worst-case performance guarantees in feed-forward networks. The classical methods (computation with (min,plus) operators) lead to upper bounds that become very loose as the network grows. A new method is presented using linear programming and mixed integer linear programming, that enables us to compute the exact performance guarantees in two cases: when the service policy is arbitrary and when the service policy is FIFO. This method can be adapted to other service policies, such as static priority. The tightness of the bounds is lost, but bounds more precise than with other methods: in fact, it is possible to combine the different approaches using more linear constraints and then take the best of them. The main drawback of this method is its computational complexity: in the case of multiplexing is arbitrary, the problem is NP-hard for general feed-forward network and the linear program has an exponential number of variables and constraints event for tandem networks in the case of the FIFO policy and we conjecture that in that case also the problem of finding the exact worst-case delay is NP-hard. The next step of this study is to mix the classical techniques and linear programming ones: in real-size networks (such as AFDX in an aircraft) thousands of flows interact, but those flows generally have a small size, 5 or 6 hops. As a consequence, one could locally apply the linear programming methods where, the characteristics of this local network are computed using more classical methods (for example, good results are obtained for studying AFDX network using grouping techniques [37]). Another open problem concerns the complexity class of obtaining tight performance bounds. If we were able to find for which classes of network and which service policies it is possible to derive tight performance bounds in polynomial time, it might give some hints about the approximability (or non-approximability) of the exact worst-case bounds in more general cases.

Third, we discussed the algorithmic issues for computing (min,plus) operators such as (min,plus)-convolution, deconvolution and sub-additive closure. Several softwares have been implemented using these algorithms (the COINC toolbox and RT@W-Pegase). A new algorithm has been recently obtained concerning the convolution in the special case of concave functions and convex functions, with application

to the simulation of a numerical scheme: when functions are discrete (defined on $\mathbb{N}$), the time-complexity is linear. Then, the convolution of a general function by a convex function can be computed by first decomposing the general function into convex and concave parts. This raises new problems, as the approximation of nearly linear parts, that alternate very fast between convex and concave parts, that makes the algorithm slow down, whereas we conjecture that this alternation is the consequence of the discretization of a smooth function that is nearly linear in this part. We gave partial answers using a tolerance parameter that enabled us to apply to such curves the algorithm of the convolution of two convex functions. As a consequence, we did not obtain rigorous results concerning the approximation. Another solution would be to directly linearize those parts of the function or try to adapt the notion of container [80].

Finally, we present an application of the arrival curves to supervise a flow and detect abnormal behaviors. The solution proposed is algorithmically very light and efficient to supervise one flow in a network. Two extensions of this work can be proposed. First the improvement of the algorithm itself. Until now, some parameters (the size of the tolerance window) have to be fixed. The use of learning algorithms might enable to avoid this problem. Moreover, due to the nature of the flows, it should be nice to give a stochastic dimension to the algorithm, enabling a few outgoing messages without recomputing a new arrival rate. Second, a network being composed of several routeurs, hence a multitude of flows, a challenge is to find how to correlate some of these flow through the algorithm presented here. The final aim is then to forcast failures in the network. In [29], we developed a preliminary model that also analyzes the occurrences of alarmes in a routeur and their correlation, but the link with the algorithm presented here has not been clarified enough. This generalized approach must be then extended to the cae of several routeur.

More perspectives

We presented only results concerning Deterministic network calculus. A first perspective would be to apply them to the stochastic Network calculus: until now, very few works make use of strict service curves (mainly, to study service policies that request it, authors use the limited case of guaranteed rate where the behavior is almost like simple service curves to make the computations valid). For example, in the book [72], the author defines the presents only one type of stochastic strict service curve when there are three types for simple service curves, all leading to different results in term of stochastic guarantees. I believe that it is possible to do better and define a stochastic version of other types of service curves. Of course, strict service curves are defined using the start of backlogged period, which make things more delicate, but variable capacity nodes might be a good candidate as its definition is simpler (in term of server) and this type of service curve is almost the same as the strict service curve. Moreover, the performance computing aspect of Stochastic Network calculus only makes use of the basic results of Deterministic Network calculus (*i.e.* (min,plus)-convolution and deconvolution). It should be possible to extend the results using mathematical programming to the stochastic case, where the deviation bounds are constraints or objectives.

A limitation of the work presented here is that, with the notable exception of the packet curves, it only concerns fluid models. Specially when we compute exact worst-case performance guarantees. Open problems are the pessimism we introduce by modeling a packet flow by a fluid one, and how it can be reduced. Linear programming and mathematical programming may not be adapted to that case. In other domains, like graph theory, problems that can be modeled by a linear program also have another, more algorithmic solution. Is it the case here? If yes, could that algorithm be used to compute exact worst case delays for packet flows? A simpler way to improve the bounds is also to adapt the recent works, like grouping flows, to the linear programming approach.

Network calculus has also recently received some attention in the field of transportation networks [57], and brings new problematics: it is an example of network with cyclic dependences between the flows and with backward control (roads have a limited capacity). Many problems are still open in that field, and in particular how to find residual service curves for one flow of vehicles under a specific service policy. It could also be a good context to try to extend the work [20] about the optimization of a flow in a network to this context.

Bibliography

[1] R. Agrawal, R.L. Cruz, C.M. Okino, and R. Rajan. A framework for adaptive service guarantees. In *Proceedings of Allerton Conf*, pages 693–702, 1998.

[2] M. Akian, S. Gaubert, and A. Lakhoua. The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. *SIAM J. Control Optim.*, 47(2):817–848, 2008.

[3] K. Altisen and M. Moy. Arrival curves for real-time calculus: the causality problem and its solutions. In *TACAS*, pages 358–372, 2010.

[4] K. Altisen and M. Moy. Causality closure for a new class of curves in real-time calculus. In *WCTT'11*, pages 3–10, 2011.

[5] F. Baccelli and B. Blaszczyszyn. *Stochastic Geometry and Wireless Networks, Volume I - Theory.* Foundations and Trends in Networking Vol. 3: No 3-4, pp 249-449. NoW Publishers, 2009.

[6] F. Baccelli and B. Blaszczyszyn. *Stochastic Geometry and Wireless Networks, Volume II - Applications.* Foundations and Trends in Networking: Vol. 4: No 1-2, pp 1-312. NoW Publishers, 2009.

[7] F. Baccelli, G. Cohen, G.Y. Olsder, and J.-P. Quadrat. *Synchronization and linearity.* Wiley, 1992.

[8] F. Baccelli, D.R. McDonald, and J. Reynier. A mean-field model for multiple TCP connections through a buffer implementing RED. *Performance Evaluation*, 49(14):77 – 97, 2002.

[9] É. Badouel, A. Bouillard, Ph. Darondeau, and J. Komenda. Residuation of tropical series: rationality issues. In *CDC-ECC'11*, 2011.

[10] P. Barta, F. Nemeth, R. Szabo, and J. Biro. End-to-end delay bound calculation in generalized processor sharing networks. In *ISCC'01*, 2001.

[11] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. Estimating the worst-case delay in fifo tandems using network calculus. In *ValueTools '08*, pages 67:1–67:10, 2008.

[12] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. Deborah: A tool for worst-case analysis of fifo tandems. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification, and Validation*, volume 6415 of *LNCS*, pages 152–168. Springer, 2010.

[13] B. Bollobás. *Random Graphs.* Studies in Advanced Mathematics. Cambridge Studies in Advanced Mathematics, 2001.

[14] A. Bouillard. Composition of service curves in network calculus. In *WCTT'11*, 2011.

[15] A. Bouillard, B. Cottenceau, B. Gaujal, L. Hardouin, S. Lagrange, L. Medhi, and É. Thierry. COINC library: a toolbox for network calculus. In *ValueTools'09*, 2009.

[16] A. Bouillard, E. Faou, and M. Zavidovique. Fast weak KAM integrators for separable hamiltonian systems. preprint, 2012.

[17] A. Bouillard, N. Farhi, and B. Gaujal. Packetization and Aggregate Scheduling. Research report RR-7685, INRIA, 2011.

[18] A. Bouillard, N. Farhi, and B. Gaujal. Packetization and Packet Curves in Network Calculus. In *Valuetools'12*, 2012.

[19] A. Bouillard, B. Gaujal, S. Lagrange, and É. Thierry. Optimal routing for end-to-end guarantees: the price of multiplexing. In *Valuetools'07*, 2007.

[20] A. Bouillard, B. Gaujal, S. Lagrange, and É. Thierry. Optimal routing for end-to-end guarantees using network calculus. *Performance Evaluation*, 65(11-12):883–906, 2008.

[21] A. Bouillard, S. Harr, and S. Rosario. Critical paths in the partial order unfolding of a stochastic petri net. In *FORMATS'09*, volume LNCS 5618, pages 43–57, 2009.

[22] A. Bouillard, C. Jard, and A. Junier. Some synchronization issues in OSPF routing. In *DCNET13*, 2013.

[23] A. Bouillard, L. Jouhet, and É. Thierry. Service curves in Network Calculus: dos and don'ts. Research Report RR-7094, INRIA, 2009.

[24] A. Bouillard, L. Jouhet, and É. Thierry. Comparison of different classes of service curves in network calculus. In *WODES'10)*, pages 316–321, 2010.

[25] A. Bouillard, L. Jouhet, and É. Thierry. Tight Performance Bounds in the Worst Case Analysis of Feed Forward Networks. In *INFOCOM'10*, 2010.

[26] A. Bouillard and A. Junier. Worst-case delay bounds with fixed priorities using network calculus. In *ValueTools'11*, 2011.

[27] A. Bouillard, A. Junier, and B. Ronot. Hidden anomaly detection in telecommunication networks. Technical report, RR-INRIA, 2012.

[28] A. Bouillard, A. Junier, and B. Ronot. Hidden anomaly detection in telecommunication networks. In *CNSM'12*, pages 82–90, 2012.

[29] A. Bouillard, A. Junier, and B. Ronot. Impact of rare alarms on event correlation. In *CNSM13*, 2013.

[30] A. Bouillard, Jouhet. L., and É. Thierry. Computation of a (min,+) multi-dimensional convolution for end-to-end performance analysis. In *Valuetools'08*, 2008.

[31] A. Bouillard, L.T.X. Phan, and S. Chakraborty. Lightweight modeling of complex state dependencies in stream processing systems. In *RTAS'09*, pages 195–204, 2009.

[32] A. Bouillard, S. Rosario, A. Benveniste, and S. Haar. Monotonicity in service orchestrations. In *ICATPN'09*, volume LNCS 5606, pages 263–282, 2009.

[33] A. Bouillard and G. Stea. Exact worst-case delay for FIFO-multiplexing tandems. In *Valuetools'12*, pages 158–167, 2012.

[34] A. Bouillard and É. Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18(1):3–49, 2008.

[35] M. Boyer and C. Fraboul. Tightening end to end delay upper bound for AFDX network calculus with rate latency FCFS servers using network calculus. In *WFCS'08*, 2008.

[36] M Boyer, J. Migge, and M. Fumey. PEGASE a robust and efficient tool for worst-case network traversal time evaluation on afdx. In *SAE Aerotech*, 2011.

[37] M. Boyer, J. Migge, and M. Fumey. Experimental assessment of timing verification techniques for afdx. In *ERTSS'12*, 2012.

[38] M. Boyer, J. Migge, and N. Navet. A simple and efficient class of functions to model arrival curve of packetised flows. In *WCTT'11*, 2011.

[39] M. Boyer, N. Navet, and M. Fumey. Experimental assessment of timing verification techniques for AFDX. In *ERTS'12*, 2012.

[40] M. Boyer, N. Navet, X. Olive, and É. Thierry. The PEGASE project: precise and scalable temporal analysis for aerospace communication systems with network calculus. In *ISOLA'10*, 2010.

[41] Y. Brenier. Un algorithme rapide pour le calcul de transofrméesde Legendre-Fenchel discrètes. *C. R. Acad. Sci. Paris. Sér. I Math.*, 308(20):587–589, 1989.

[42] E. Brugallé. Some aspects of tropical geometry. *Eur. Math. Soc. Newsl.*, 83:23–28, 2012.

[43] S. Chakraborty, L.T.X. Phan, and P.S. Thiagarajan. Event count automata: A state-based model for stream processing systems. In *RTSS*, pages 87–98, 2005.

[44] C.-S. Chang. Stability, queue length and delay of deterministic and stochastic queueing networks. *IEEE Transactions on Automatic Control*, 39:913–931, 1994.

[45] C.-S. Chang. *Performance Guarantees in Communication Networks*. TNCS, Springer-Verlag, 2000.

[46] C.-S. Chang, D.-S. Lee, and Y.-S. Jou. Load balanced Birkhoff-von Neumann switches. In *IEEE Workshop on High Performance Switching and Routing*, pages 276–280, 2001.

[47] C.-S. Chang, D.-S. Lee, and Y.-S. Jou. Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering. *Computer Communications*, 25(6):611–622, 2002.

[48] C.-S. Chang, D.-S. Lee, and C.-M. Lien. Load balanced Birkhoff–von Neumann switches, part II: Multi-stage buffering. *Computer Communications*, 25(6):623–634, 2002.

[49] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

[50] G. Contreras, R. Iturriaga, and H. Sanchez-Morgado. Weak solutions of the Hamilton-Jacobi equation for time periodic Lagrangians. *preprint*, 2000.

[51] R.L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[52] R.L. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[53] R.L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on selected areas in communication*, 13:1048–1056, 1995.

[54] M. Draief and L. Massouli. *Epidemics and Rumours in Complex Networks*. London Mathematical Society Lecture Note Series. Cambridge, 2010.

[55] N.G. Duffield, K.K. Ramakrishnan, and A.R. Reibman. Save: An algorithm for smoothed adaptive video over explicit rate networks. *IEEE/ACM Transactions on Networking*, 6:717–728, 1998.

[56] K.R. Duffy. Mean field markov models of wireless local area networks. *Markov Processes and Related Fields*, 16(2):295–328, 2010.

[57] N. Farhi, H. Haj-Salem, and J.-P. Lebacque. Algebraic approach for performance bound calculus on transportation networks (road network calculus). *Transportation Research Record 2014*, 2013.

[58] A. Fathi. Weak KAM Theorem in Lagrangian Dynamics, preliminary version, Pisa, 16 février 2005.

[59] T. Ferrandiz, F. Frances, and C. Fraboul. A network calculus model for spacewire networks. In *RTCSA'11*, volume 1, pages 295–299, 2011.

[60] M. Fidler. A survey of deterministic and stochastic service curve models in the network calculus. *IEEE Communications Surveys & Tutorials*, 12(1), 2010.

[61] M. Fidler and V. Sander. A parameter based admission control for differentiated services networks. *Comput. Netw.*, 44(4):463–479, 2004.

[62] F. Frances, C. Fraboul, and J. Grieu. Using network calculus to optimize AFDX network. In *ERTS'06*, 2006.

[63] M. Franceschetti and R Meester. *Random networks for Communication: from statistical physics to information systems*. Series in Statistical and probabilistic mathematics. Cambridge, 2008.

[64] L.T.X. Gangadharan, D.and Phan, S. Chakraborty, R. Zimmermann, and I. Lee. Video quality driven buffer sizing via frame drops. In *RTCSA*, pages 319–328, 2011.

[65] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[66] N. Gast, B. Gaujal, and J.-Y. Le Boudec. Mean field for markov decision processes: From discrete to continuous optimization. *IEEE Transactions on Automatic Control*, 57(9):2266–2280, 2012.

[67] S. Gaubert and J. Mairesse. *Idempotency*, chapter Task Resource Models and (max,+) Automata. Isaac Newton Institute, 1998.

[68] S. Gaubert and J. Mairesse. Modeling and analysis of timed petri nets using heaps of pieces. *IEEE Trans. Autom. Control*, 44(4):683–697, 1999.

[69] N. Gollan, F.A. Zdarsky, I. Martinovic, and J.B. Schmitt. The DISCO network calculator. In *Proc. of MMB'2008*, 2008.

[70] Moy. J. RFC 2328 OSPF v2, 1998.

[71] G.-S. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.*, 126:202–228, 1996.

[72] Y. Jiang and Y. Liu. *Stochastic Network Calculus*. Springer, 2008.

[73] L. Jouhet. *Algorithmique du Network Calculus*. thèse, Ecole normale supérieure de lyon, 2012.

[74] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard. Optimism due to serialization in the trajectory approach for switched ethernet networks. In *Proc. of the 7th Junior Researcher Workshop on Real-Time Computing (JRWRTC 2013)*, Sophia Antipolis, France, October 16-18 2013.

[75] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard. Optimistic problems in the trajectory approach in fifo context. In *Proc. of the 18th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA 2013)*, 2013.

[76] D. Kirsten and S. Lombardy. Deciding unambiguity and sequentiality of polynomially ambiguous min-plus automata. In *STACS*, pages 589–600, 2009.

[77] I. Klimann, S. Lombardy, J. Mairesse, and C. Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.*, 327(3):349–373, 2004.

[78] J.-Y. Le Boudec, D. McDonald, and J. Mundinger. A generic mean field convergence result for systems of interacting objects. In *QEST'07*, pages 3–18, 2007.

[79] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume LNCS 2050. Springer-Verlag, 2001. revised version 4, May 10, 2004.

[80] E. Le Corronc, B. Cottenceau, and L. Hardouin. Container of (min,+)-linear systems. *Journal of Discrete Event Dynamic Systems*, 2012.

[81] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks. *Performance Evaluation*, 63(9-10):956–987, 2006.

[82] L. Lenzini, E. Mingozzi, and G. Stea. End-to-end delay bounds in FIFO-multiplexing tandems. In *Valuetools'07*, 2007.

[83] L. Lenzini, E. Mingozzi, and G. Stea. A methodology for computing end-to-end delay bounds in FIFO-multiplexing tandems. *Performance Evaluation*, 65(11-12):922–943, 2008.

[84] D.V. Lindley. The theory of queues with a single server. *Mathematical Proceedings of the Cambridge Philosophical Society*, 48:277–289, 4 1952.

[85] J. Little. A proof for the queuing formula: L= λ w. *Operations Research*, 9(3):pp. 383–387, 1961.

[86] S. Martin and P. Minet. Worst case end-to-end response times of flows scheduled with FP/FIFO. In *ICN/ICONS/MCL*, page 54, 2006.

[87] J. M. McManus and K. W. Ross. Video-on-demand over ATM: Constant-rate transmission and transport. *IEEE J.Sel. A. Commun.*, 14(6):1087–1098, September 2006.

[88] F. Nemeth, P. Barta, R. Szabo, and J. Biro. Network internal traffic characterization and end-to-end delay bound calculus for generalized processor sharing scheduling discipline. *Computer Networks*, 48(6):910–940, 2005.

[89] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[90] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple-node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, 1994.

[91] L.T.X. Phan, S. Chakraborty, and P.S. Thiagarajan. A multi-mode real-time calculus. In *RTSS*, pages 59–69, 2008.

[92] L.T.X. Phan, S. Chakraborty, P.S. Thiagarajan, and L. Thiele. Composing functional and state-based performance models for analyzing heterogeneous real-time systems. In *RTSS*, 2007.

[93] Florence Plateau. *Modèle n-synchrone pour la programmation de réseaux de Kahn à mémoire bornée*. PhD thesis, Université Paris-Sud 11, 2010.

[94] Y. Qian, Z. Lu, and W. Dou. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *NoCS 2009*, pages 44–53, 2009.

[95] H. Schioler, J.J. Jessen, J. Dalsgaard, and K.G. Larsen. Network calculus for real time analysis of embedded systems with cyclic task dependencies. In *Proc. of Computers and Their Applications*, 2005.

[96] H. Schioler, H.-P. Schwefel, and M.B. Hansen. CyNC: a MATLAB/Simulink toolbox for network calculus. In *Valuetools'07*, 2007.

[97] J.B. Schmitt, N. Gollan, S. Bondorf, and I. Martinovic. Pay Bursts Only Once Holds for (Some) Non-FIFO Systems. In *INFOCOM'11*, 2011.

[98] J.B. Schmitt and F.A. Zdarsky. The DISCO network calculator: a toolbox for worst case analysis. In *Valuetools'06*, 2006.

[99] J.B. Schmitt, F.A. Zdarsky, and M. Fidler. Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch ... In *INFOCOM'08*, 2008.

[100] J.B. Schmitt, F.A. Zdarsky, and I Martinovic. Improving performance bounds in feed-forward networks by paying multiplexing only once. In *Proc. of MMB'2008*, 2008.

[101] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.

[102] W.M. Sofack and M. Boyer. Non preemptive static priority with network calculus. In *ETFA*, pages 1–8, 2011.

[103] W.M. Sofack and M. Boyer. Non preemptive static priority with network calculus: Enhancement. In *MMB/DFT*, pages 258–272, 2012.

[104] L. Thiele and N. Stoimenov. Modular performance analysis of cyclic dataflow graphs. In *Proceedings of the seventh ACM international conference on Embedded software*, pages 127–136. ACM, 2009.

[105] E. Wandeler. *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. PhD thesis, ETH Zurich, September 2006.

[106] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006.

[107] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis - a case study. *Software Tools for Technology Transfer*, 8(6):649–667, 2006.

Abstract

This document presents some results obtained in the field of network calculus, a theory based on the (min,plus) algebra and whose aim is to compute worst-case performance bounds in communication networks. This theory models flows circulating in a network and the service offered by the network elements by cumulative functions and those functions are abstracted by enveloped on which the computations are performed. Several aspects are addressed.

A first part is devoted to the clarification and the improvements of the performance bounds computed using this theory: the different types of service curves and the relation between them are clarified; a new operator of packet curves introduced, in order to describe the packet sizes the same way as the flows; and we improve the way of computing worst-case performance bounds, that is classically based on the (min,plus) operators, by introducing linear programs that compute the exact worst-case performances in some cases and improve the bounds in the other cases.

The second part presents some examples other application of the results first developed for networks calculus: algorithms of convolution of (min,plus) functions have received a lot of attention by the network calculus community in order to compute bounds efficiently. We show here an example of use for approximating the numerical solution of the Hamilton-Jacobi equation. Another example is to use the concept of arrival curve to supervise a flow. This is done with simple algorithm that can follow the evolution of the behavior of a flow.

Résumé

Ce mémoire présente quelques résultats obtenus dans la domaine du Network calculus, une théorie basée sur l'algèbre (min,plus) et dont le but est de calculer des bornes supérieures des performances dans des réseaux de communication. Les flots de données circulant dans un réseaux ainsi que les éléments de réseaux sont modélisés par des functions cumulées, elles-mêmes abstraites par des enveloppes sur lesquelles les calculs sont effectivement effectués. Plusieurs problèmes sont abordés.

Une première partie est dédiée à la clarification de certains concepts et à l'amélioration des bornes calculées : différents types de courbes de service sont identifés et les relations entre eux clarifiés ; un nouvel opérateur pour décrire les paquets est introduit, similaire aux courbes d'arrivées (décrivant les flots de données) ; une modélisation par programmation linéaire est introduite pour améliorer les bornes obtenues via des opérateurs (min,plus) et obtenir des bornes exactes dans certains cas.

La seconde partie présente des exemples d'utilisation de résultats d'abord développés pour le network calculus pour d'autre domaines : l'utilisation de la convolution (min,plus) pour approximer numériquement les solutions de l'équation de Hamilton-Jacobi et l'utilisation du concept de courbe d'arrivée pour la supervision d'un flot de données, à l'aide d'un algorithme simple qui permet de suivre l'évolution d'un flot de données.